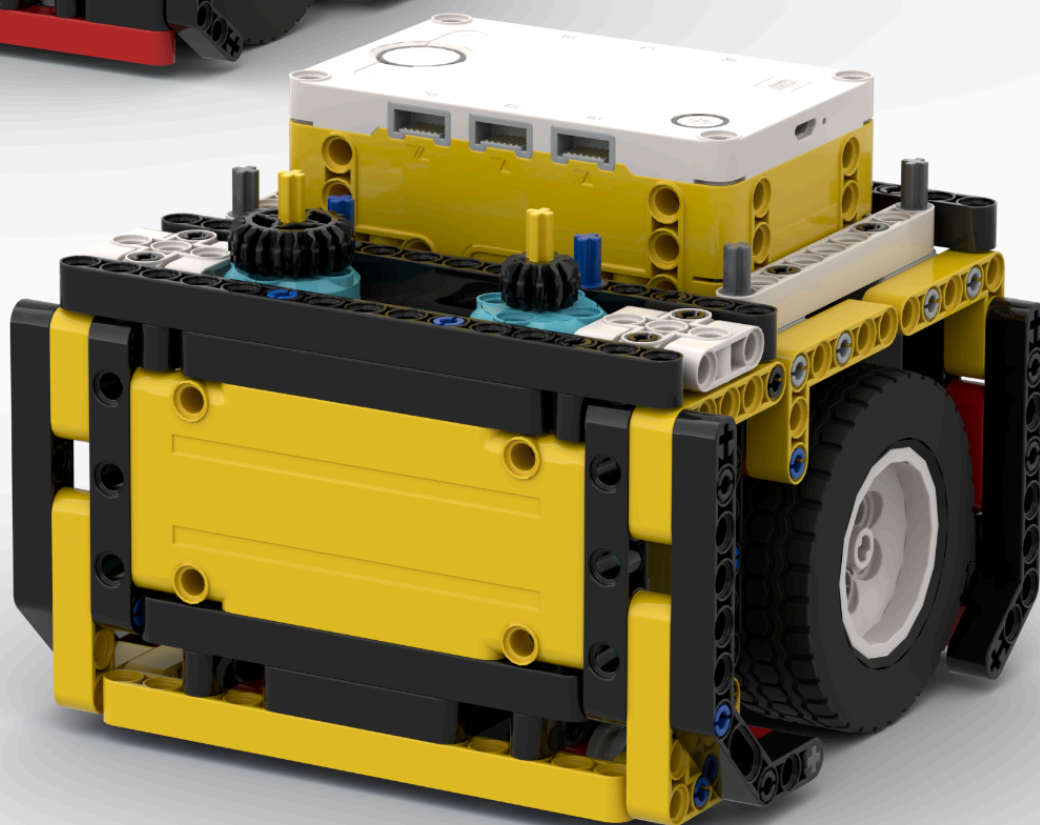
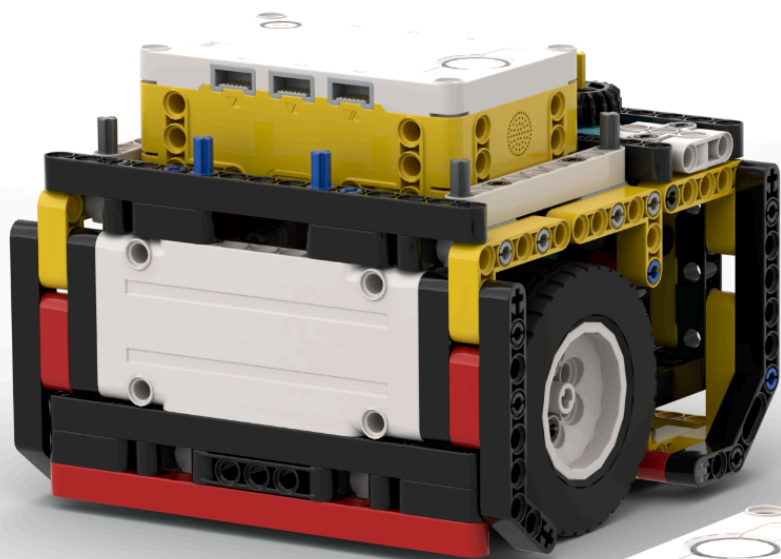




**FIRST
LEGO
LEAGUE**

KARGO

Fiche technique du robot

Table des matières

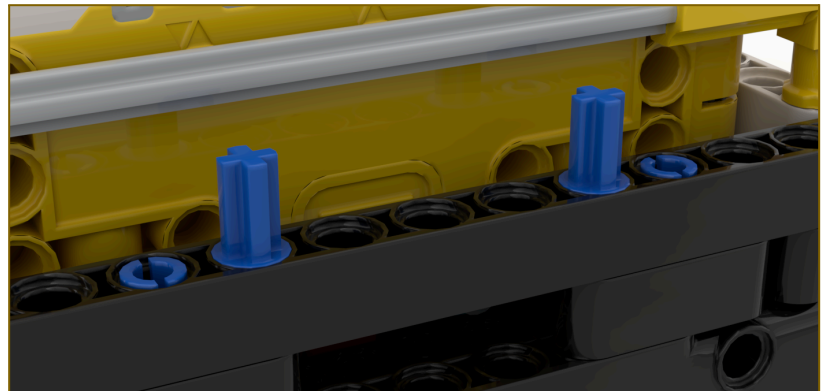
Le robot.....	3
Vue technique du robot.....	4
Analyse du terrain.....	5
Choix des sorties.....	6
Remue-méninge des sorties.....	6
Regrouper par zone.....	6
Répartition des tâches.....	7
Tâches à accomplir.....	7
Composition des équipes.....	7
Processus itératif.....	8
Exemple d'itérations.....	8
Changement de roues.....	9
Changement de mécanisme.....	9
Programmation - Blocs vs Python.....	10
Programmation par bloc.....	10
Programmation Python (pybricks).....	10
Programmation - Boîte à outils.....	11
Paramètres du robot et variables.....	11
Vérifier le voltage de la batterie.....	13
Boussole.....	14
Déplacement en CM/MM.....	15
Déplacement en secondes/millisecondes.....	16
Tourner sur une seule roue.....	17
Tourner avec les deux roues.....	19
Type de mécanisme.....	21
Actif.....	21
Passif.....	21
Détachable (non-utilisé cette année).....	21
Indépendant.....	21
Aide au pilotage.....	22
Capteur de couleur.....	22
Gabarit de départ.....	22
Alignement mécanique.....	23
Sortie A.....	24
Mécanisme - Pelle.....	25
Mécanisme - Cage.....	26
Programmation.....	27
Sortie B.....	29
Mécanisme - Proue.....	30
Mécanisme - Gobe coffre.....	31
Programmation.....	32
Sortie C.....	34
Mécanisme - Couteau Suisse.....	35
Mécanisme - Bocal.....	36
Mécanisme - La chute.....	37
Programmation.....	38
Sortie D.....	40
Mécanisme - Fourche.....	40
Programmation.....	42
Sortie E.....	44
Mécanisme - Glissade.....	45
Mécanisme - Guide.....	46
Programmation.....	47

Le robot

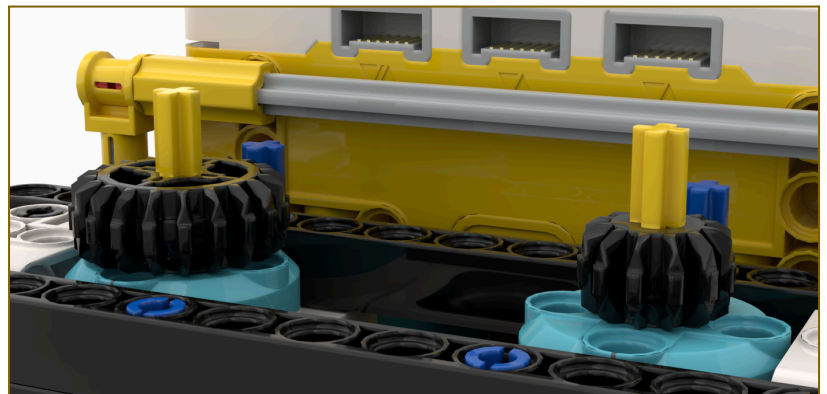
Dimension	
Largeur	15 unités
Profondeur	15 unités
Hauteur (avant)	10 unités
Hauteur (arrière)	13 unités
Circonférence des roues	5,5 cm



Caractéristiques
Il est fait de parois lisses des quatres côtés afin d'éviter de se prendre dans les modules et de pouvoir s'appuyer sur des murs ou des modèles de mission.
Il y a deux moteurs avec des engrenages de différentes grandeurs qui permettent des ratios de force/vitesse différents pour nos modules
Il y a des points d'ancrage à l'avant et à l'arrière.
Sur l'arrière et l'avant du robot, on a intégré des fentes pour les modules détachables.

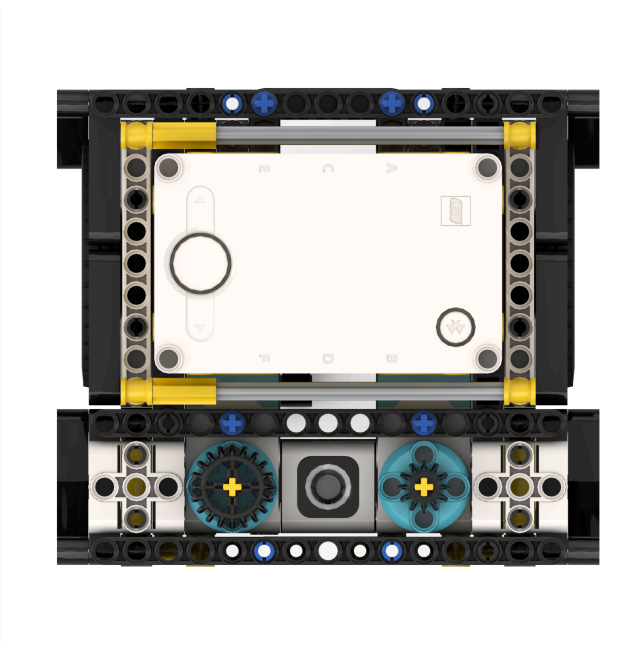
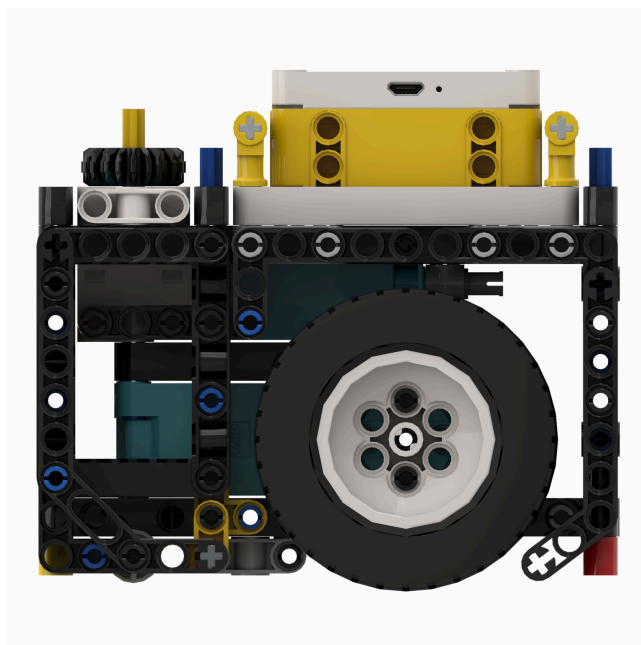
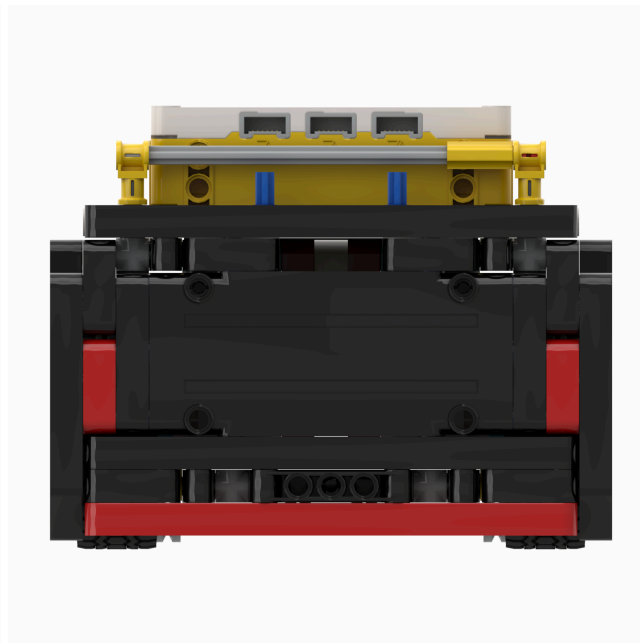
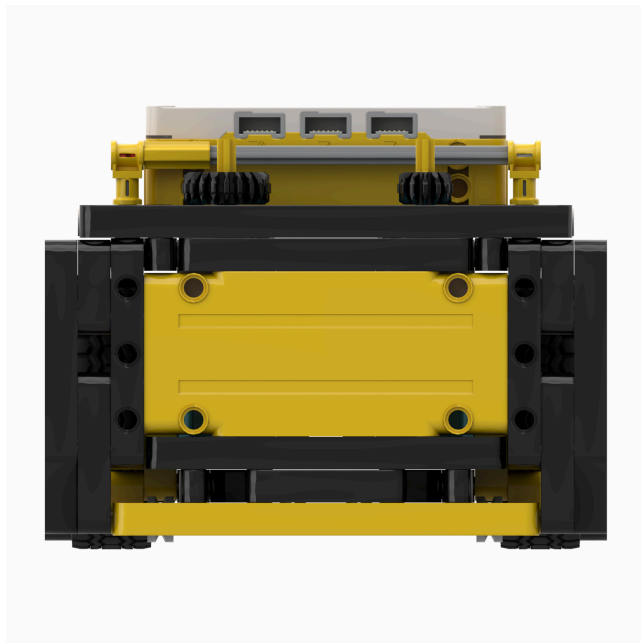


Moteurs	Port	Description
Moyen	A	Module gauche
Moyen	E	Module droit
Moyen	B	Roue gauche
Moyen	F	Roue droite



Capteurs	Utilisé?	Port	Raisons
Capteur gyroscopique	Oui	Intégré	Pour effectuer des virages précis et corriger notre trajectoire.
Capteur de couleur	Oui	C	Permet de réduire le travail des techniciens en lançant le bon programme.
Capteur de distance	Non	-	Nous avons choisi d'utiliser le gyroscope et la circonférence des roues pour la précision de nos déplacements. De plus, nos modules pourraient interférer avec le capteur.
Capteur tactile	Non	-	Nous n'avons pas trouvé de façon avantageuse d'utiliser ce capteur.

Vue technique du robot



Analyse du terrain

Mission	Difficulté estimée	Points max.	Type de mécanisme estimé / Notes
M00 - Inspection	Facile	20	Aucun
M01 - Pépinière de corail	Moyen	50	Actif
M02 - Requin	Facile	30	Passif
M03 - Récif corallien	Facile	35	Passif
M04 - Plongeur sou- marin	Moyen	40	Actif
M05 - Poisson pêcheur	Facile	30	Passif
M06 - Lever le mât	Facile	30	Passif
M07 - Le trésor du kraken	Difficile	20	Passif
M08 - Habitat artificiel	Difficile	40	Actif
M09 - Rencontre inattendue	Facile	30	Passif
M10 - Envoyez le sous marin	Facile	40	Actif
M11 - Découverte du sonar	Difficile	30	Actif
M12 - Nourrissez la baleine	Difficile	50	Passif et actif
M13 - Modifiez les couloirs de navigation	Difficile	20	Actif
M14 - Collecte d'échantillons	Moyen	55	Actif et passif
M15 - Navire de recherche	Moyen	50	Actif et indépendant
M16 - Jetons de précision	Facile	50	Aucun



Choix des sorties

Remue-méninge des sorties

En sous-équipes, nous avons noté sur papier des sorties possibles et par la suite, nous avons analysé les différentes sorties qui ont été proposées. En équipe, nous avons fait le choix de cinq sorties retenues.

Regrouper par zone

Nous avons regroupé des missions (entre deux et quatre) qui étaient proches les unes des autres pour créer des sorties optimales. Cela nous permet de faire un maximum de missions dans le temps alloué. Nous avons aussi réparti les missions selon les zones de lancement qu'elles soient à gauche ou à droite.



Répartition des tâches

Tâches à accomplir

	Sous-équipes			
	1	2	3	4
Conception	Sortie C	Sortie D	Sortie B	Sortie E
Programmation	Sortie A	Sortie C	Sortie D	Sortie B
Plan 3D	Sortie B	Sortie A	Sortie C	Sortie D
Documentation	Sortie D	Sortie B	Sortie A	Sortie C

Composition des équipes

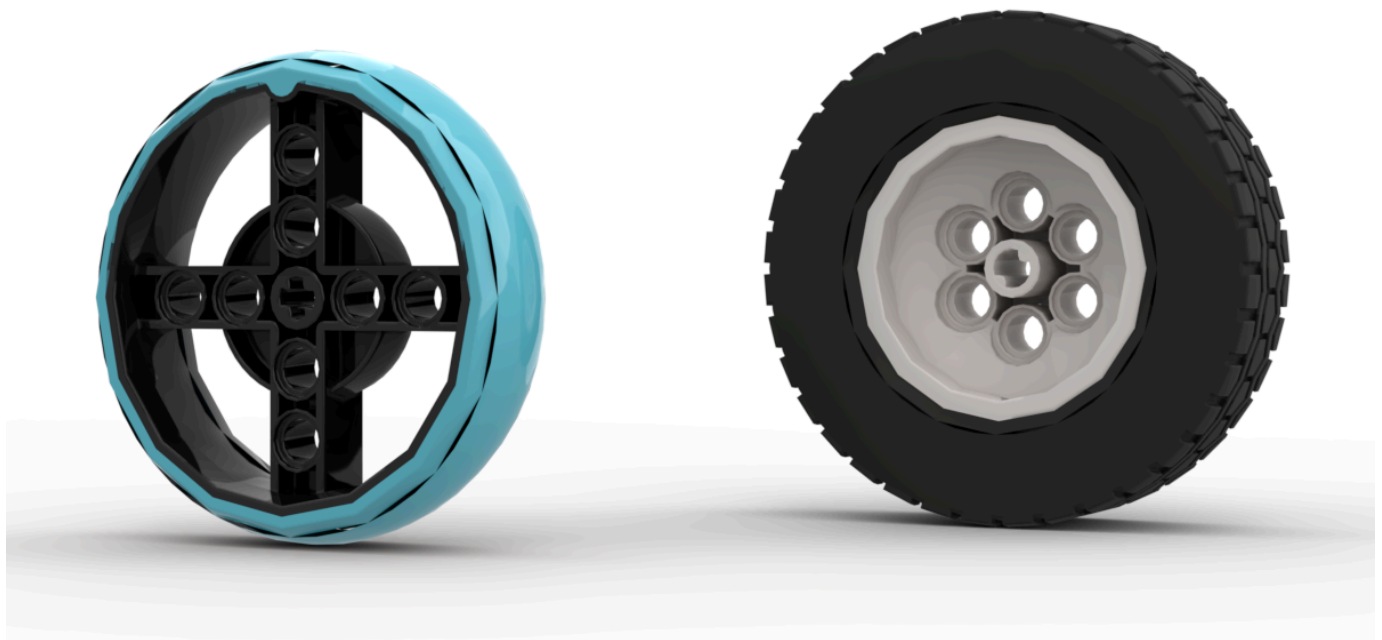
Les sous-équipes ont été composées de façon à répartir les forces. Nous avons combiné des membres forts en programmation ou en conception avec des membres qui débutent dans le domaine. Ceci a contribué à échanger des connaissances et de faire en sorte que tout le monde soit un peu plus au même niveau. Nous avons également placé un membre qui avait déjà fait une saison dans chacune des sous-équipes.

Processus itératif



Planifier	Nous analysons ce que nous devons faire et nous planifions ce qui doit être réalisé.
Concevoir	Nous construisons le mécanisme selon ce qui est requis et que l'on a observé.
Essayer	Suite à notre conception, nous effectuons des tests afin d'observer le résultat.
Apprendre	Nous trouvons les problèmes et nous apprenons de nos erreurs. Nous modifions les mécanismes pour éliminer les problèmes observés.

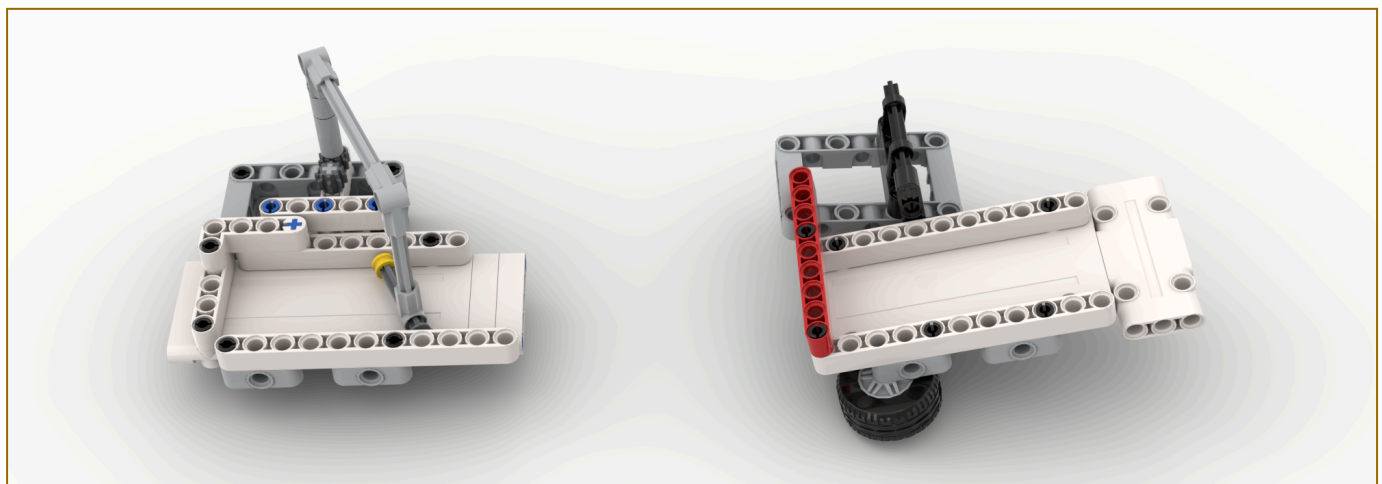
Exemple d'itérations



Changement de roues

Nous avons changé les roues de notre robot car les roues bleues du Spike ont une petite surface de contact sur le sol. Ceci faisait en sorte que le robot avait peu d'adhérence et glissait parfois sur le tapis. Une autre raison est que les roues accumulent la poussière plus rapidement ce qui cause les dérapages. Bref, les nouvelles roues ont une plus grande surface et sont moins affectées par la poussière.

Changement de mécanisme



Programmation - Blocs vs Python

Programmation par bloc

Avantages
Facile à comprendre et à concevoir; le texte sur les blocs rend la programmation plus facile et claire.
Ce type de langage est plus accessible pour les nouveaux programmeurs.
Difficile de faire des erreurs et de faire planter le programme.

Inconvénients
Le langage en blocs peut se montrer plutôt imprécis lorsqu'il s'agit de virages en degrés et de distances en centimètres.
Le langage en blocs utilise les centimètres pour ses distances, au lieu des millimètres.
On ne peut pas accélérer ou décélérer facilement en utilisant le gyroscope.

Programmation Python (pybricks)

Avantages
Le langage de programmation Python offre nombreuses autres fonctionnalités que les blocs, tel que les virages dynamiques, l'accélération et le voltage de la batterie,
Python est plus précis que le langage en blocs, il mesure ses distances en millimètres, le temps en millisecondes et il a aussi une plus petite marge d'erreur.
Au lieu d'estimer, on peut définir le nombre de dents des engrenages pour rendre le programme encore plus précis.

Inconvénients
Le langage Python peut être difficile à comprendre à première vue, avec toute la ponctuation apparemment inutile!
La moindre petite erreur fait planter le programme au complet.

Programmation - Boîte à outils

La boîte à outils nous permet de programmer rapidement le déplacement du robot. Elle permet aussi de centraliser le tout ce qui fait en sorte qu'on peut modifier le comportement du robot en un seul endroit.

Paramètres du robot et variables

Description

Les paramètres du robot nous servent à définir toutes les variables du robot. Cela comprend le diamètre des roues et tous les ports avec les modules que nous avons utilisés. Les variables de vitesse sont utilisées pour changer la vitesse partout dans le programme sans devoir changer les chiffres à chaque endroit. Lorsqu'on programme pour se déplacer, on choisit l'une des vitesses prédéfinies.

```
#=====
#Paramètre du robot
#=====

diametre_roue = 62.4    # -> en millimètres
distance_roue = 97      # -> en millimètres

roue_gauche = Motor(Port.D,Direction.COUNTERCLOCKWISE)
roue_droite = Motor(Port.F,Direction.CLOCKWISE)

capteur_couleur = ColorSensor(Port.B)

voltage_maximum = 8372

#=====
#Variables
#=====

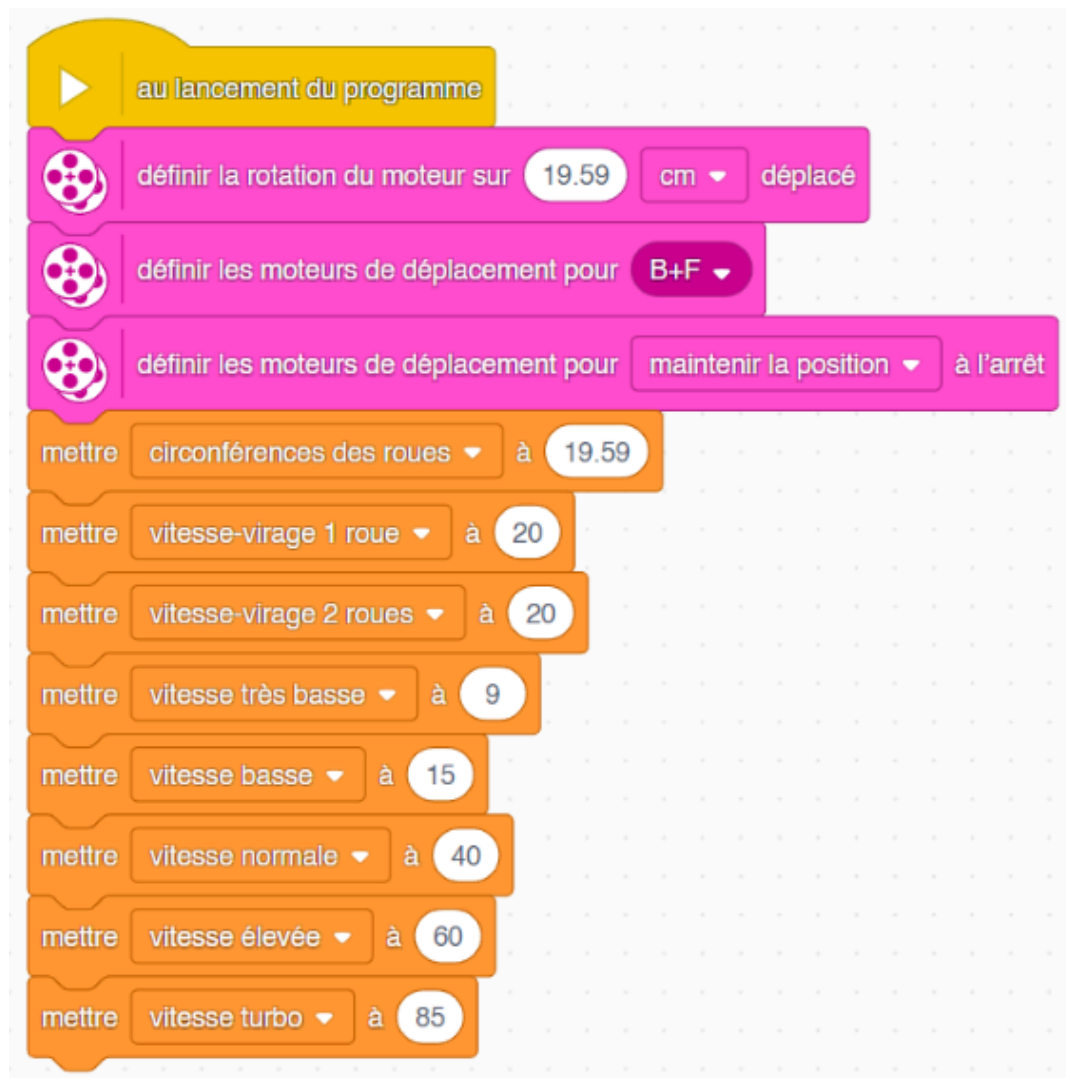
vitesse_tres_basse = 80
vitesse_basse = 200
vitesse_normale = 400
vitesse_elevee = 650
vitesse_tres_elevee = 900

vitesse_acceleration = 600

vitesse_virage = 280
vitesse_acceleration_virage = 450
vitesse_virage_minimum = 120

#=====
#Initialisation du hub et du robot
#=====

hub = PrimeHub()
robot = DriveBase(roue_gauche,roue_droite,diametre_roue,distance_roue)
```



Vérifier le voltage de la batterie

Description

Cette fonction permet de savoir le voltage de la batterie et son pourcentage.

```
#=====
#FONCTION : VOLTAGE DE LA BATTERIE
#=====

def verifierBatterie():
    etat_batterie = ((hub.battery.voltage() / voltage_maximum *100))

    print("*** Le voltage est à "+str(etat_batterie)+"% (" + str(hub.battery.voltage())+")")
    print("*** Le voltage lorsque pleinement chargé est aux alentours de "+str(voltage_maximum))
```

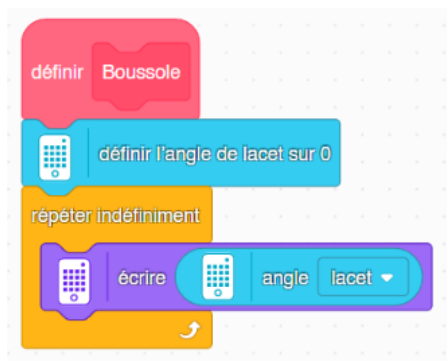
Boussole

Description

La fonction boussole sert à déterminer l'angle nécessaire pour tourner.

```
#=====
#FONCTION : BOUSSOLE
#=====

def boussole():
    robot.reset()
    while True:
        hub.display.text(str(robot.angle()))
```



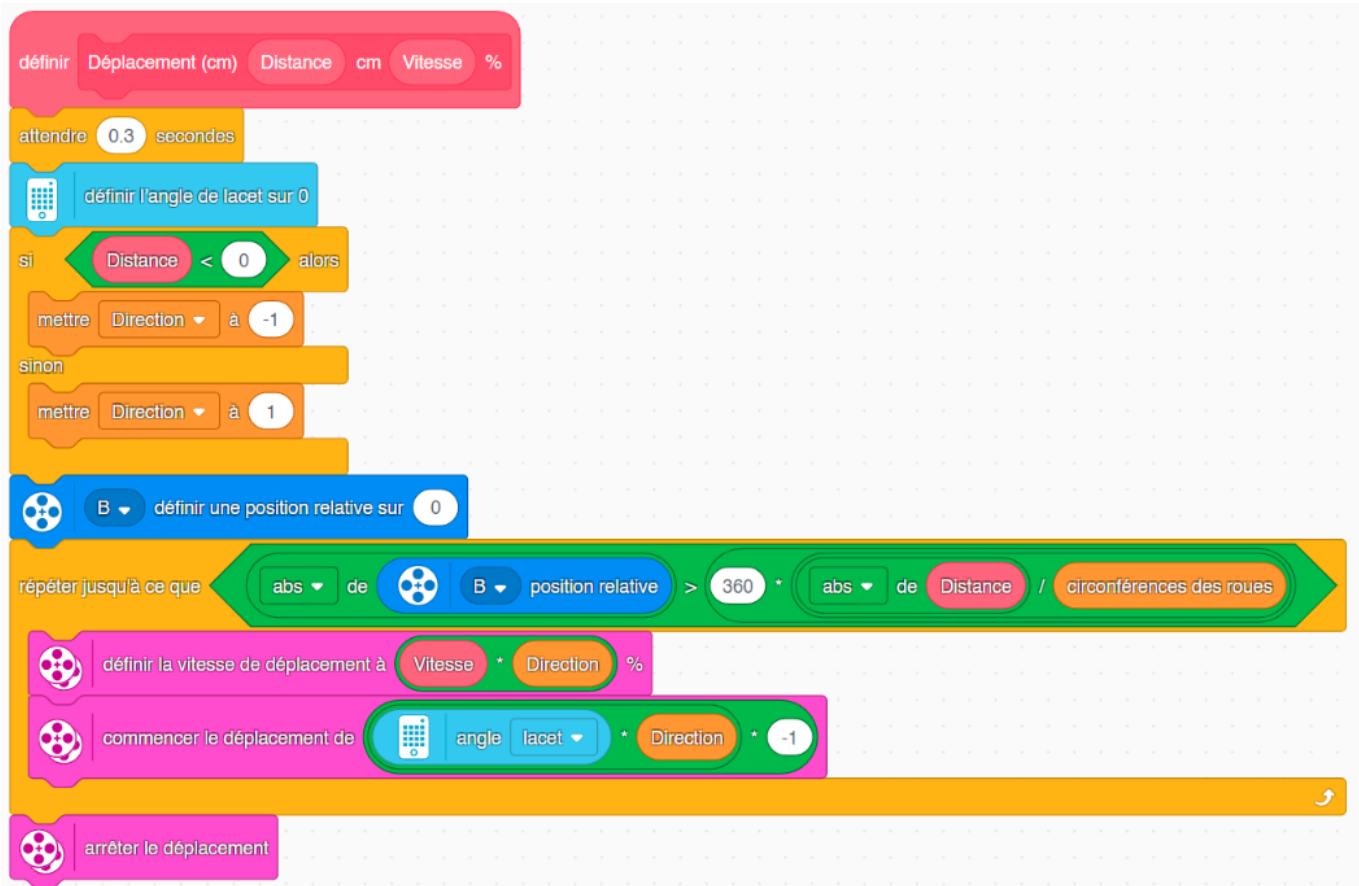
Déplacement en CM/MM

Description

Cette fonction permet de se déplacer en centimètre selon la circonférence des roues. Le gyroscope nous permet de corriger la trajectoire et de garder une ligne droite.

```
#=====
#FONCTION : DÉPLACEMENT EN MILLIMÈTRES
#=====

def deplacerDistance(distance,vitesse):
    robot.use_gyro(True)
    robot.settings(vitesse,vitesse_acceleration,vitesse_virage,vitesse_acceleration_virage)
    robot.straight(distance)
    robot.stop()
```

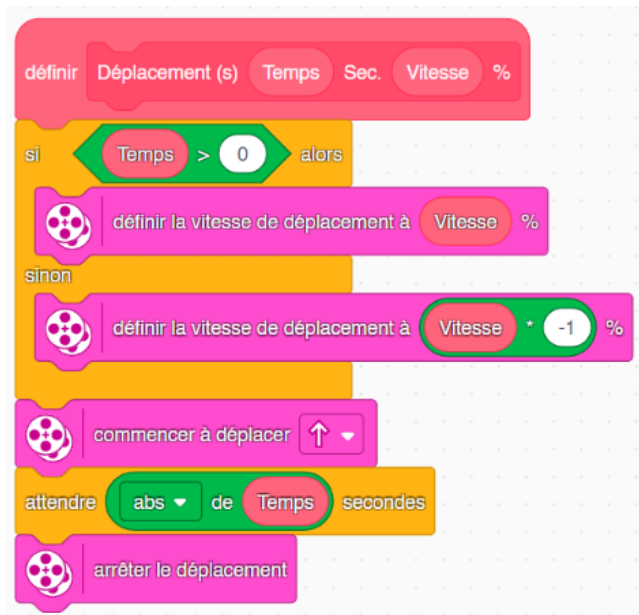


Déplacement en secondes/millisecondes

Description

Cette fonction permet de se déplacer selon une limite de temps. Elle permet de s'accoter sur les modules ou les murs présents sur la table de jeux sans bloquer la programmation.

```
#####  
#FONCTION : DEPLACEMENT EN MILLISECONDES  
#####  
  
def deplacerTemps(temps,vitesse):  
    wait(200)  
    robot.use_gyro(False)  
    if(temps < 0):  
        direction = -1  
    else:  
        direction = 1  
    robot.drive((vitesse * direction),0)  
    wait(abs(temps))  
    robot.stop()  
    wait(300)
```



Tourner sur une seule roue

Description

Cette fonction permet de tourner avec une seule roue en bloquant la roue opposée. Le gyro sert à calculer l'angle de rotation avec une variable de direction.

```
#=====
#FONCTION : TOURNER SUR 1 ROUES
#=====

def tournerUneRoue(angleVirage, roue):
    robot.use_gyro(True)
    #On réinitialise le robot/gyro
    robot.reset()

    #On vérifie la direction du virage (avant = 1, arrière = -1)
    if(angleVirage > 0):
        direction = 1
    else:
        direction = -1

    #On bloque les roues avant le virage
    roue_gauche.hold()
    roue_droite.hold()

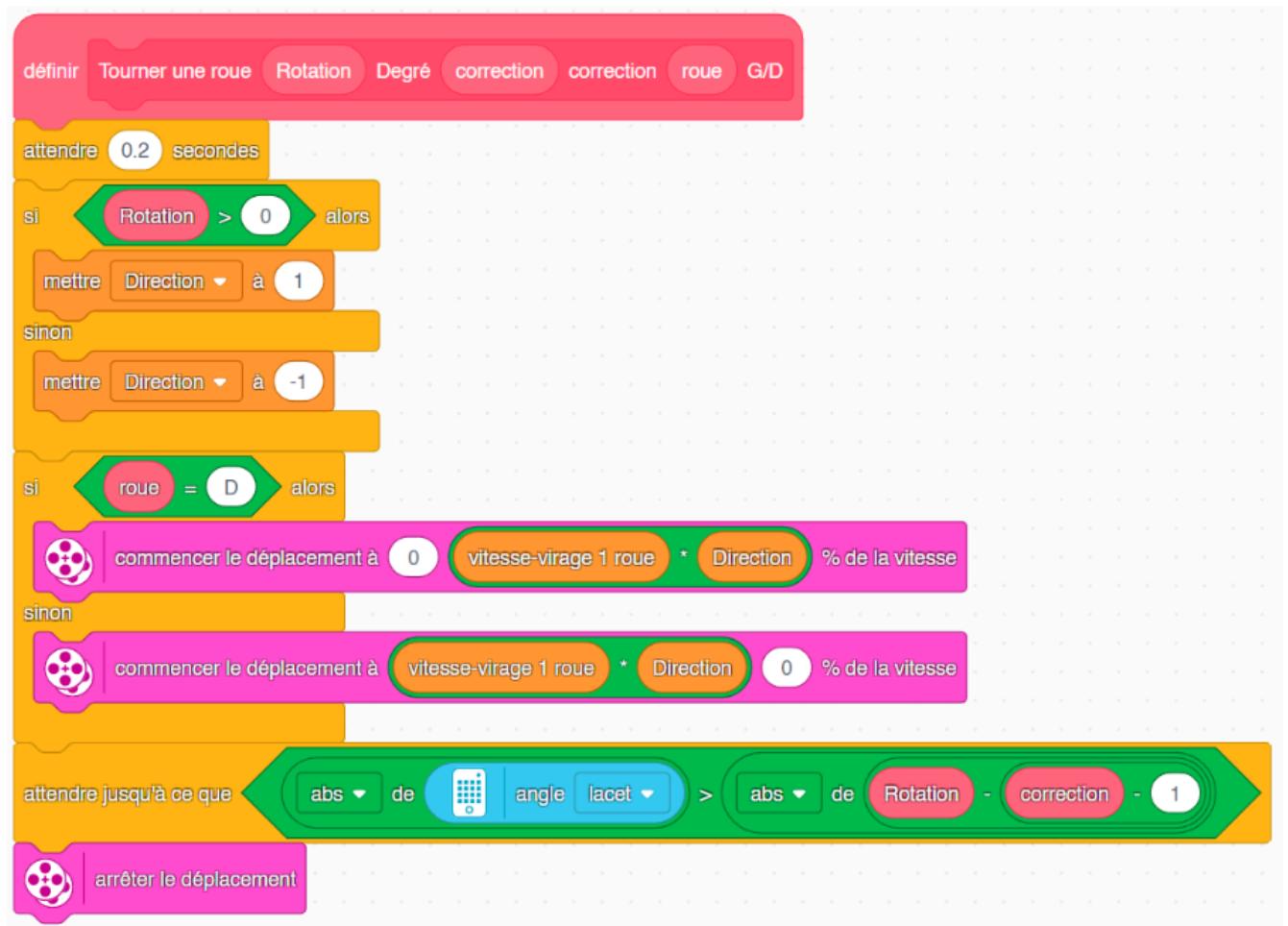
    while(abs(robot.angle()) < abs(angleVirage)):
        #Le coefficient détermine le % du virage qui est en cours
        coefficient = ((abs(angleVirage) - abs(robot.angle())) / abs(angleVirage))

        #On définit la vitesse du virage. La vitesse ralentit plus on avance vers la fin du virage
        #La vitesse minimum évite que le robot s'arrête par manque de force.
        vitesse_roue = ((vitesse_virage * 2 * coefficient) + vitesse_virage_minimum) * direction

        #On détermine ici quelle roue bouge.
        if(roue == "gauche"):
            roue_gauche.run(vitesse_roue)

        else:
            roue_droite.run(vitesse_roue)

    #On bloque les roues après le virage
    roue_gauche.hold()
    roue_droite.hold()
```

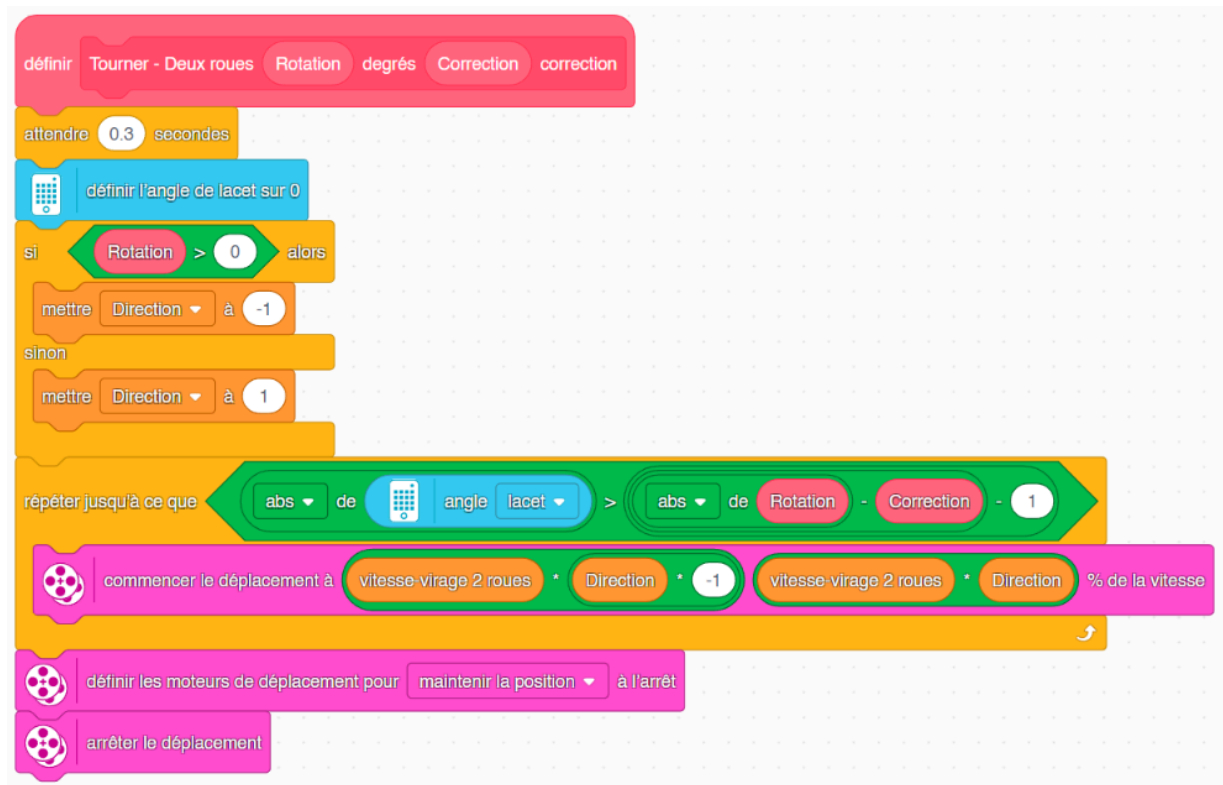


Tourner avec les deux roues

Description

Cette fonction permet de tourner avec une roue dans les deux directions, ce qui permet de tourner dans un espace plus restreint.

```
#####  
#FONCTION : TOURNER SUR 2 ROUES  
#####  
  
def tournerDeuxRoues(angleVirage):  
    robot.use_gyro(True)  
    robot.reset()  
  
    if(angleVirage > 0):  
        direction = 1  
    else:  
        direction = -1  
  
    while(abs(robot.angle()) < abs(angleVirage)):  
        coefficient = ((abs(angleVirage) - abs(robot.angle())) / abs(angleVirage))  
  
        vitesse_syncro = ((vitesse_virage * coefficient) + vitesse_virage_minimum) * direction  
  
        roue_gauche.run(vitesse_syncro)  
        roue_droite.run(vitesse_syncro * -1)  
  
    roue_gauche.stop()  
    roue_droite.stop()
```



Type de mécanisme

Actif

Description	Avantages	Inconvénients
C'est un mécanisme qui utilise les moteurs du robot.	• Ça permet de faire des mécanismes qui nécessitent des mouvements plus complexes. • Avec ce type de mécanisme, on peut également soulever et déplacer certaines parties de la table ou du robot.	• Les engrenages peuvent sauter si la force demandée est trop élevée. • On est limité à 2 moteurs par sortie. • Les mécanismes avec les moteurs sont généralement plus gros à cause des systèmes d'engrenages.

Passif

Description	Avantages	Inconvénients
C'est un mécanisme qui n'utilise aucun moteur et qui sert du déplacement du robot pour fonctionner.	• N'utilise aucun moteur. • En moyenne plus constant qu'un mécanisme actif. • sauve de l'espace.	• On ne peut pas faire de mouvement complexe. • La solution est souvent plus simple avec un mécanisme actif.

Détachable (non-utilisé cette année)

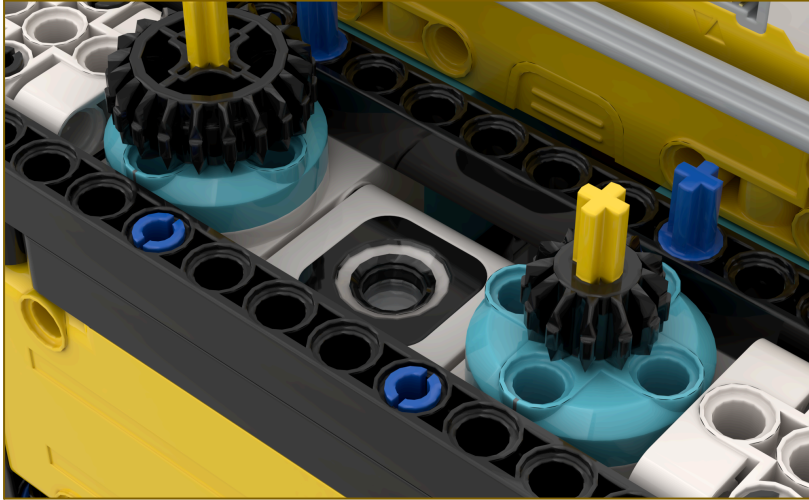
Description	Avantages	Inconvénients
C'est un module qui se détache de façon volontaire du robot lors d'une sortie	• Après avoir déposer le module en place, le robot devient plus petit,	• Après avoir été libéré le module prend de l'espace sur la table de jeu et peut nuire si le robot essaie de repasser à cet endroit.

Indépendant

Description	Avantages	Inconvénients
C'est un module qui n'est pas attaché au robot et qui est autonome dans ses mouvements.	• Comme il est indépendant, on ne perd pas de temps car son action est parallèle à celle du robot.	• Prend de l'espace dans la zone de lancement. • Seulement pour les missions qui sont proches de la zone de lancement.

Aide au pilotage

Capteur de couleur



Le capteur de couleur nous permet d'être plus rapide dans nos changements de modules, car lorsqu'on dépose un module, il nous reste seulement à appuyer sur le bouton de démarrage pour lancer le robot. La barre de couleur située sur le module permet de démarrer le bon programme. Cette façon de faire permet d'économiser beaucoup de temps lors de la préparation du robot.

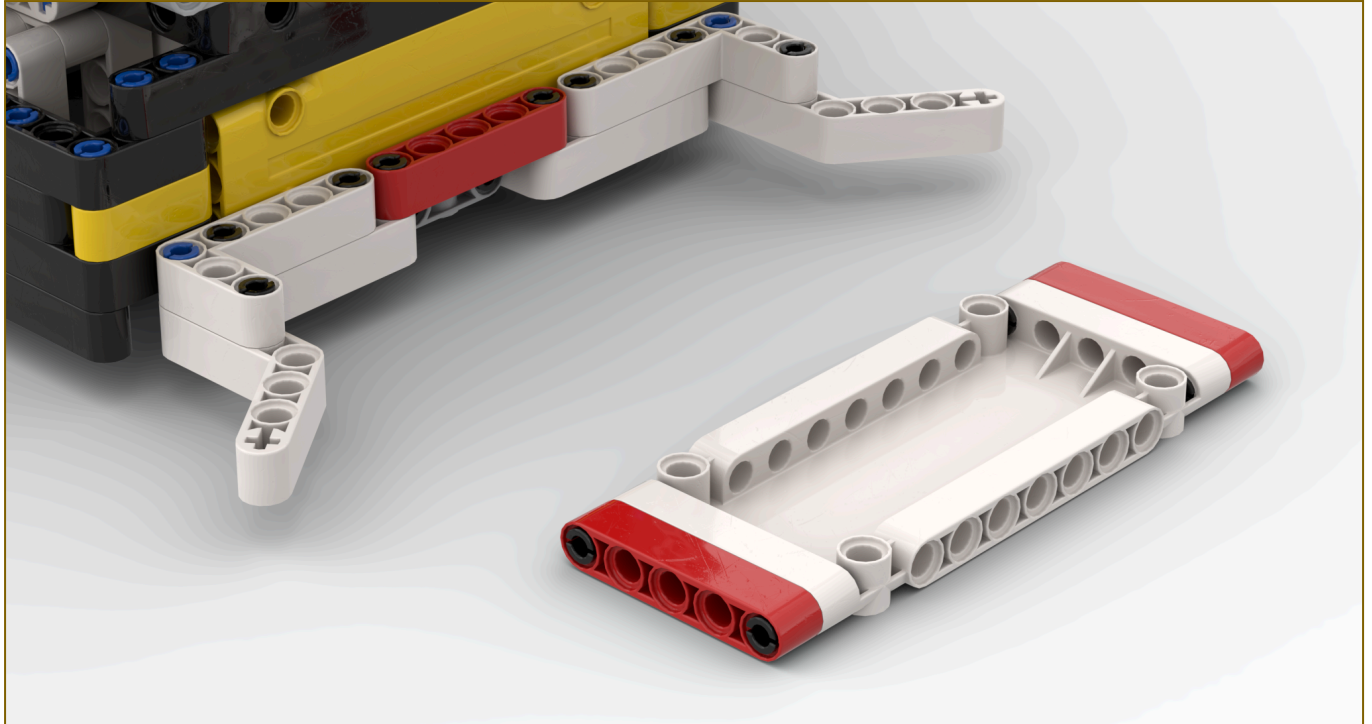


Gabarit de départ



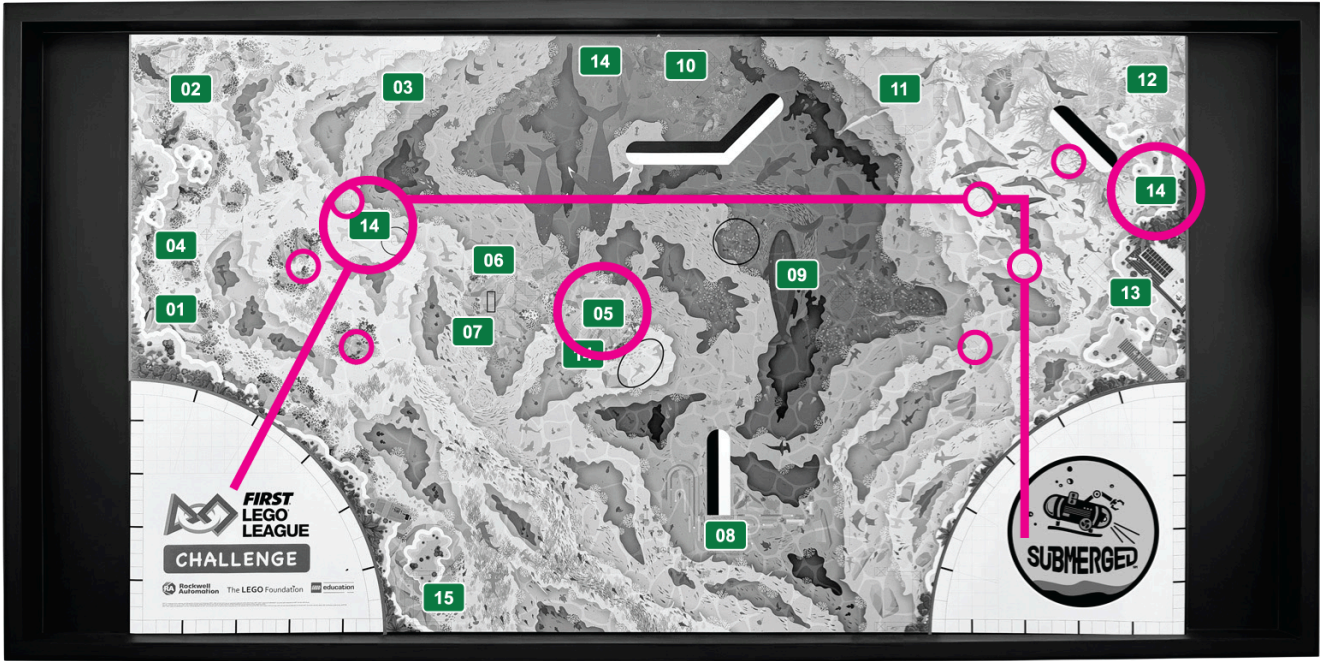
Les gabarits servent à bien positionner le robot au point de départ. Cela peut s'avérer utile pour éviter des problèmes de positionnement. C'est un avantage tactique sur la table lors des compétitions car ceci permet de sauver du temps et cela baisse les chances d'avoir des problèmes. L'angle des gabarits varie selon le besoin et la position voulue.

Alignement mécanique



L'alignement mécanique sert à mieux s'aligner sur des objets. Le mécanisme sur le robot a la forme d'un entonnoir plat et fait un travail exceptionnel. Quand le robot n'est pas tout à fait sur la cible, le mécanisme ramène le robot droit et parfait. L'avantage de ce mécanisme est que peu importe l'erreur accumulée, le robot va s'enligner.

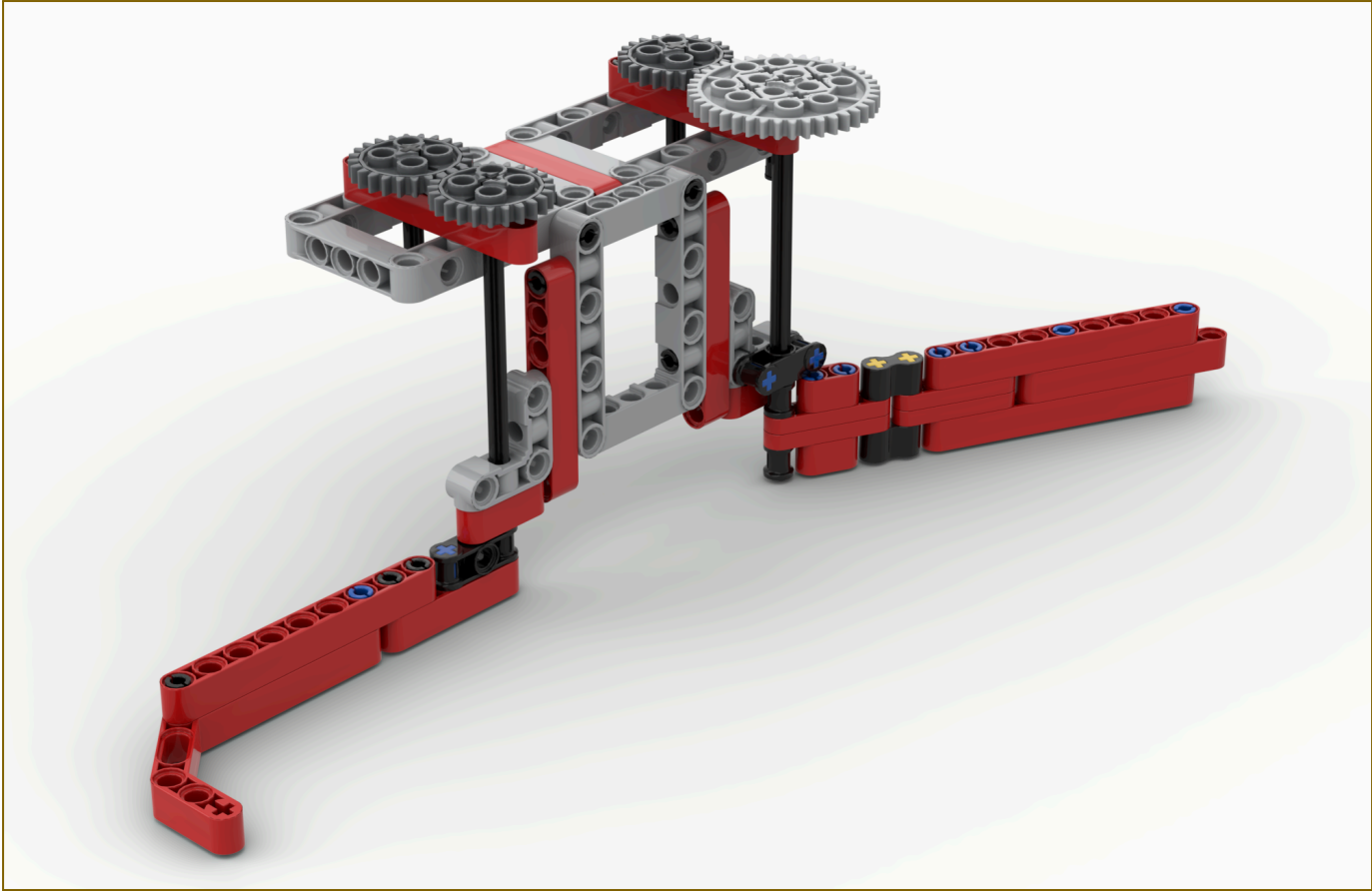
Sortie A



Mission	Pointage	Type de mécanisme
M14 - Collecte d'échantillons	15 pts	Passif
M12 - Nourrissez la baleine	Ramassage des krills	Passif et actif
M05 - Poisson pêcheur	30 pts	Actif
M03 - Récif corallien	Ramassage des récifs	Passif et actif
Total	45 pts	

Temps
15 secondes

Mécanisme - Pelle



Type	Actif
Position	Avant du robot

Mécanisme avec engrenage	Ratio	Effet
Pelle gauche	1:1	La vitesse et la force demeurent inchangées.
Pelle droite	1.67:1	La vitesse est diminué de 1,67 fois. La force est augmentée de 1,67 fois.

Problème rencontré	Solution apportée
La pelle gauche n'était pas assez longue.	Nous avons allongé la pelle gauche pour atteindre le poisson globe.

Mécanisme - Cage



Type	Passif
Position	Arrière du robot

Mécanisme avec engrenage	Ratio	Effet
Aucun	-	

Problème rencontré	Solution apportée
Le plancton touchait le fond de la cage alors le robot tremblait et changeait de direction.	Nous avons allongé la cage pour que le plancton puisse rentrer au complet dedans.
Les tiges du module étaient trop longues.	Essayer plusieurs longueurs de tiges jusqu'à trouver la bonne.

Programmation

```
def sortieA():

    moteur_gauche = Motor(Port.A, positive_direction=Direction.CLOCKWISE, gears=[[20,12], [40,24]],
reset_angle=True)
    moteur_droit = Motor(Port.E, positive_direction=Direction.CLOCKWISE, gears=[[12,20], [24,24]],
reset_angle=True)

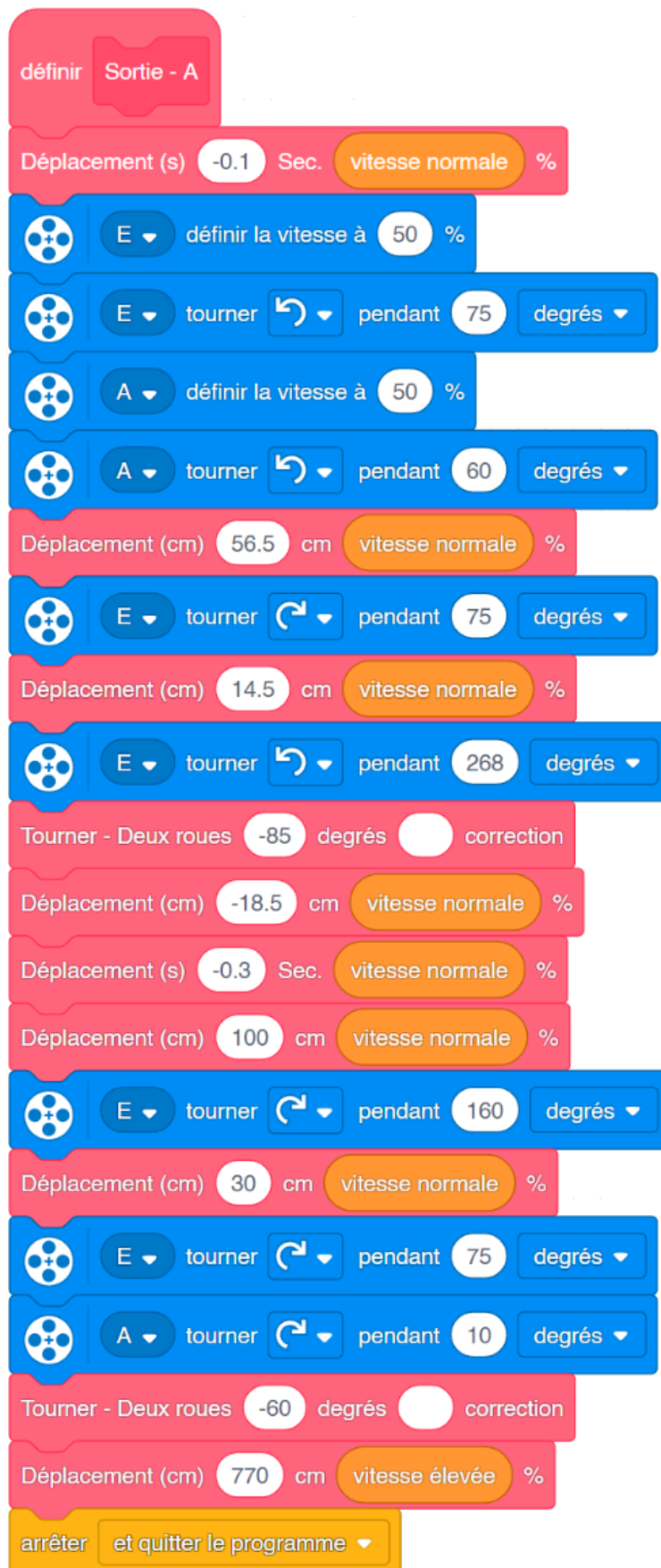
    #On initialise Le robot
    deplacerTemps(-200, vitesse_normale)

    #On ramasse La première section de Krill et de corail
    moteur_gauche.run_angle(300,-135)
    moteur_droit.run_angle(300, -45)
    deplacerDistance(565,vitesse_elevee)
    moteur_droit.run_angle(300, 45)

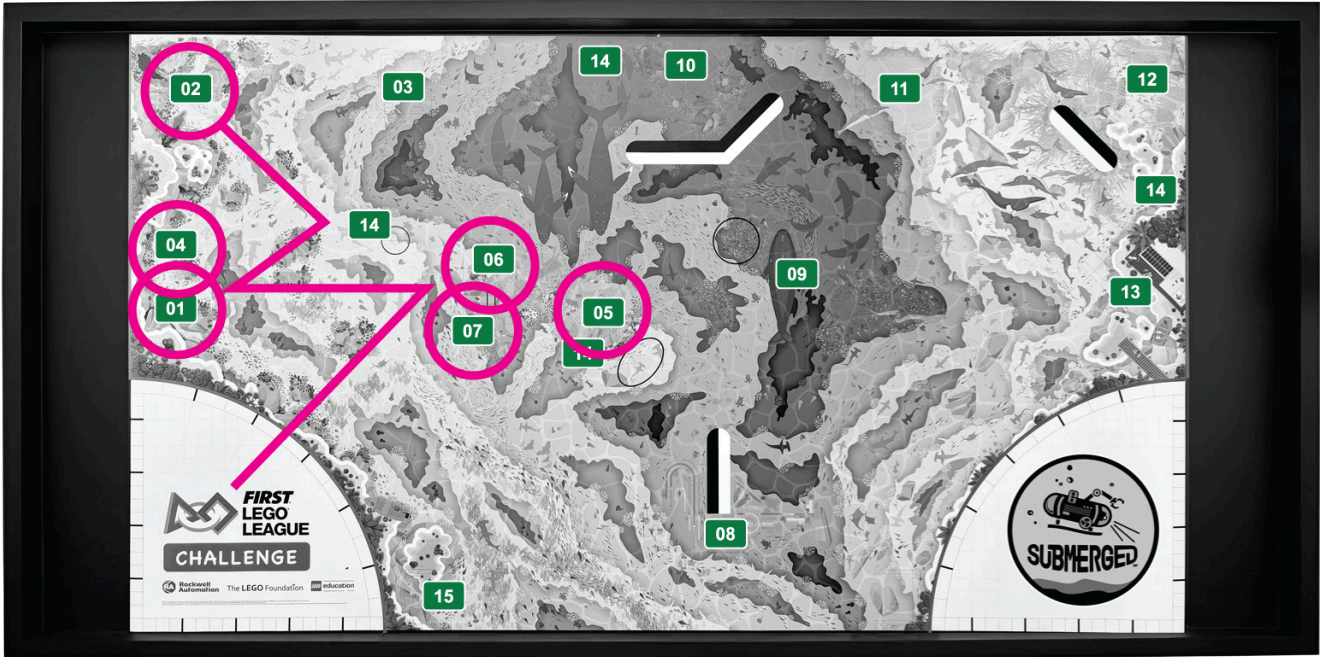
    #M14 - Collecte d'échantillons (Plancton)
    deplacerDistance(165,vitesse_elevee)
    robot.stop()
    moteur_droit.run_angle(300, -160)
    tournerDeuxRoues(-90)
    deplacerDistance(-185, vitesse_normale)
    deplacerTemps(-300, vitesse_normale)

    #M05 - Poisson pêcheur
    deplacerDistance(1000, vitesse_normale)

    #On ramasse Le restant des krills et des coraux
    #M14 - Collecte d'échantillons (Eau)
    moteur_droit.run_angle(300,45)
    deplacerDistance(300,vitesse_normale)
    moteur_droit.run_angle(300, 45)
    moteur_gauche.run_angle(300, 45)
    tournerDeuxRoues(-60)
    deplacerDistance(770, vitesse_elevee)
```



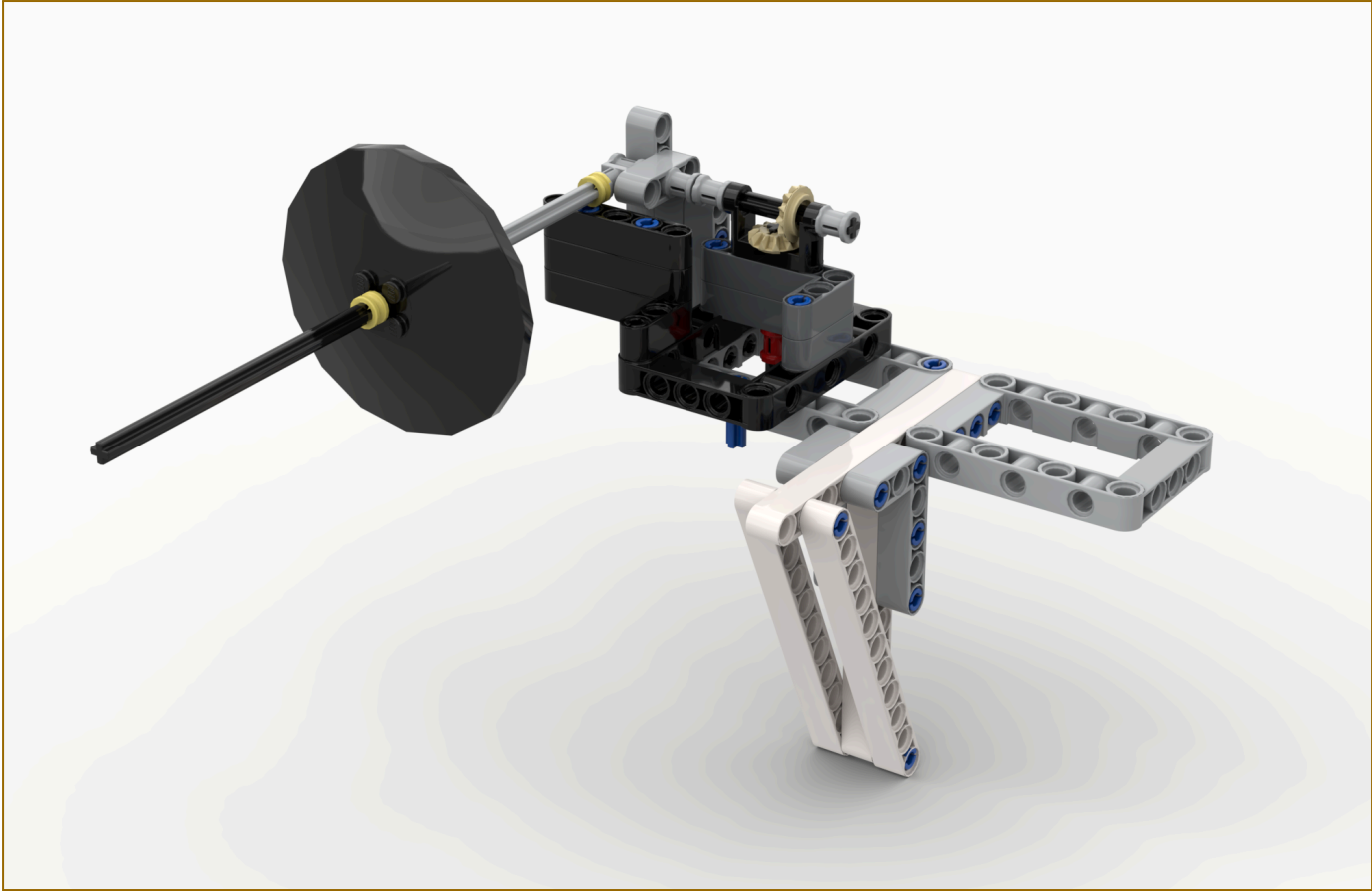
Sortie B



Mission	Pointage	Type de mécanisme
M07 - Le trésor du kraken	20 pts	Actif
M06 - Lever le mât	30 pts	Passif
M01 - Pépinière de corail	20 pts	Actif
M04 - Plongeur sous marin	20 pts	Actif
M02 - Requin	20 pts	Passif
Total	110 pts	

Temps
17 secondes

Mécanisme - Proue



Type	Actif et passif
Position	Avant

Mécanisme avec engrenage	Ratio	Effet
Baguette pour chercher le plongeur	1.67:1	La vitesse est diminué de 1,67 fois. La force est augmentée de 1,67 fois.

Problème rencontré	Solution apportée
La baguette projetait le plongeur en l’air	Nous avons réduits la force de la baguette par programmation
L’angle avant n’avait pas le bon angle.	Nous avons essayer jusqu’à trouver le bon

Mécanisme - Gobe coffre



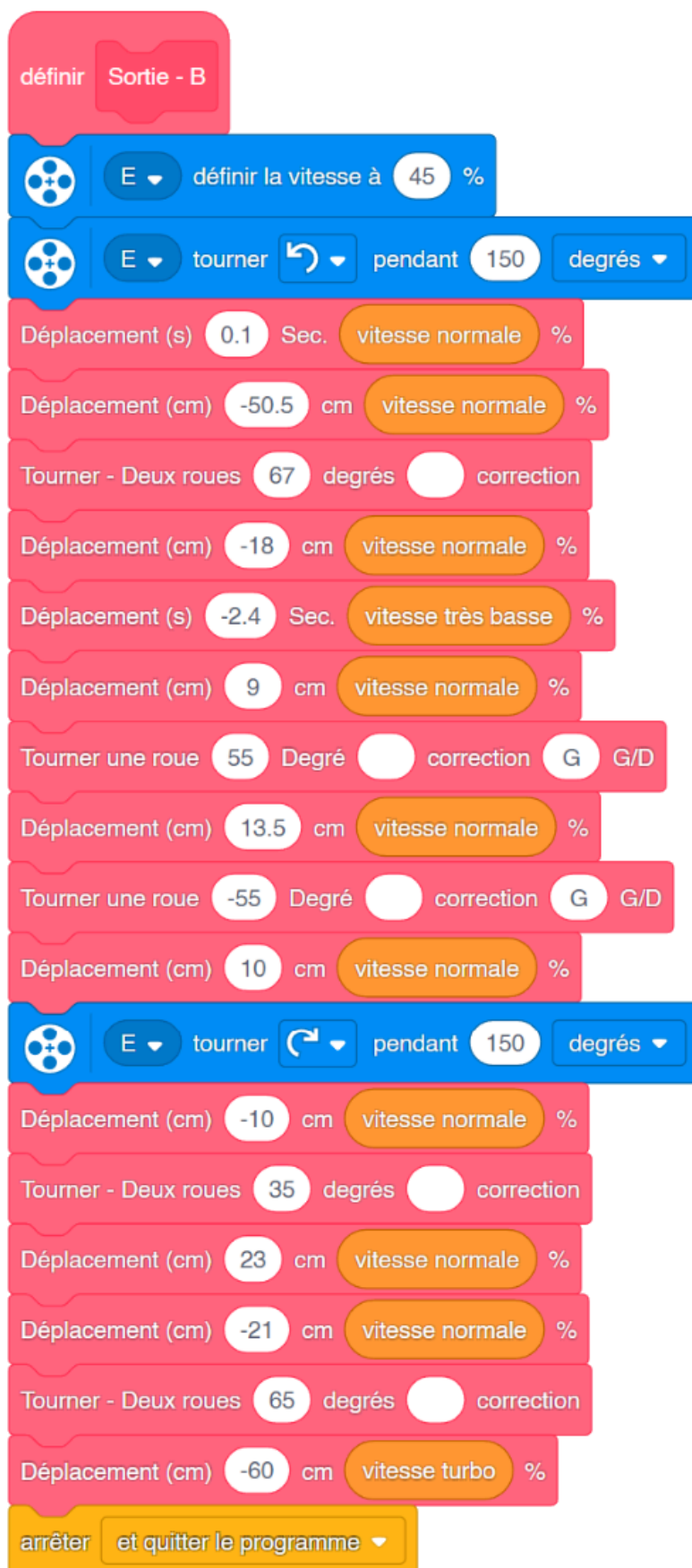
Type	Passif
Position	Arrière

Mécanisme avec engrenage	Ratio	Effet
-	-	-

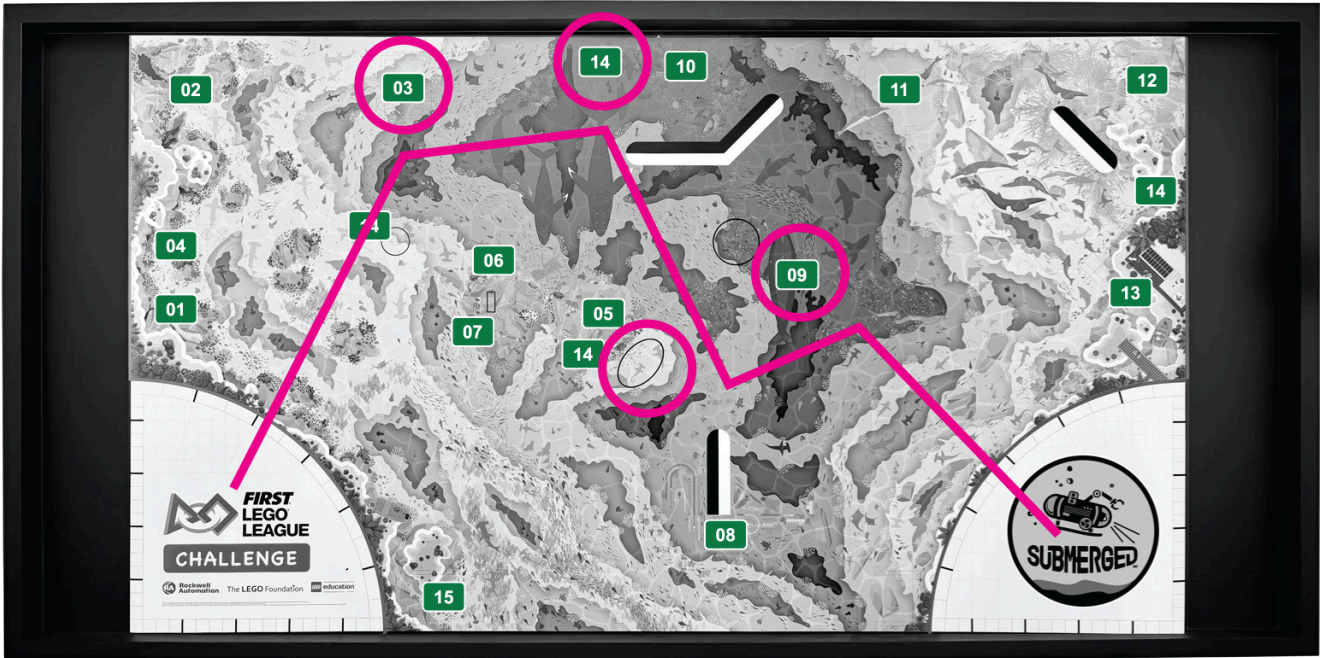
Problème rencontré	Solution apportée
Difficultés à prendre le coffre de manière efficace	Dispositif qui agrippe le coffre par au-dessus, au lieu d'autour.
Les dents du module ne fonctionnait pas de façon prévue	Essayer différents types de dents
La pente pour lever le mât restait coincé.	On a retiré le mécanisme car il était superflu.

Programmation

```
#####  
#SORTIE B  
#####  
  
def sortieB():  
  
    moteur_droit = Motor(Port.E, positive_direction=Direction.CLOCKWISE, gears=[12,20], reset_angle=True)  
  
    #On initialise le robot  
    moteur_droit.run_angle(360,-90)  
    deplacerTemps(100,vitesse_normale)  
  
    #M06 - Lever le mât & M07 - Le trésor du kraken  
    deplacerDistance(-505,vitesse_normale)  
    tournerDeuxRoues(72)  
    deplacerDistance(-180,vitesse_normale)  
    deplacerTemps(-1200,vitesse_tres_basse)  
  
    #M01 - Pépinière de corail & #M04 - Plongeur sous-marin  
    deplacerDistance(90,vitesse_normale)  
    tournerUneRoue(60,"gauche")  
    deplacerDistance(165,vitesse_normale)  
    tournerUneRoue(-60,"gauche")  
    deplacerDistance(100,vitesse_basse)  
    moteur_droit.run_angle(90,80)  
  
    #M02 - Requin  
    deplacerDistance(-100,vitesse_normale)  
    tournerDeuxRoues(40)  
    deplacerDistance(230,vitesse_normale)  
  
    #Retour du robot  
    deplacerDistance(-210,vitesse_normale)  
    tournerDeuxRoues(70)  
    deplacerDistance(-600,vitesse_tres_elevee)
```

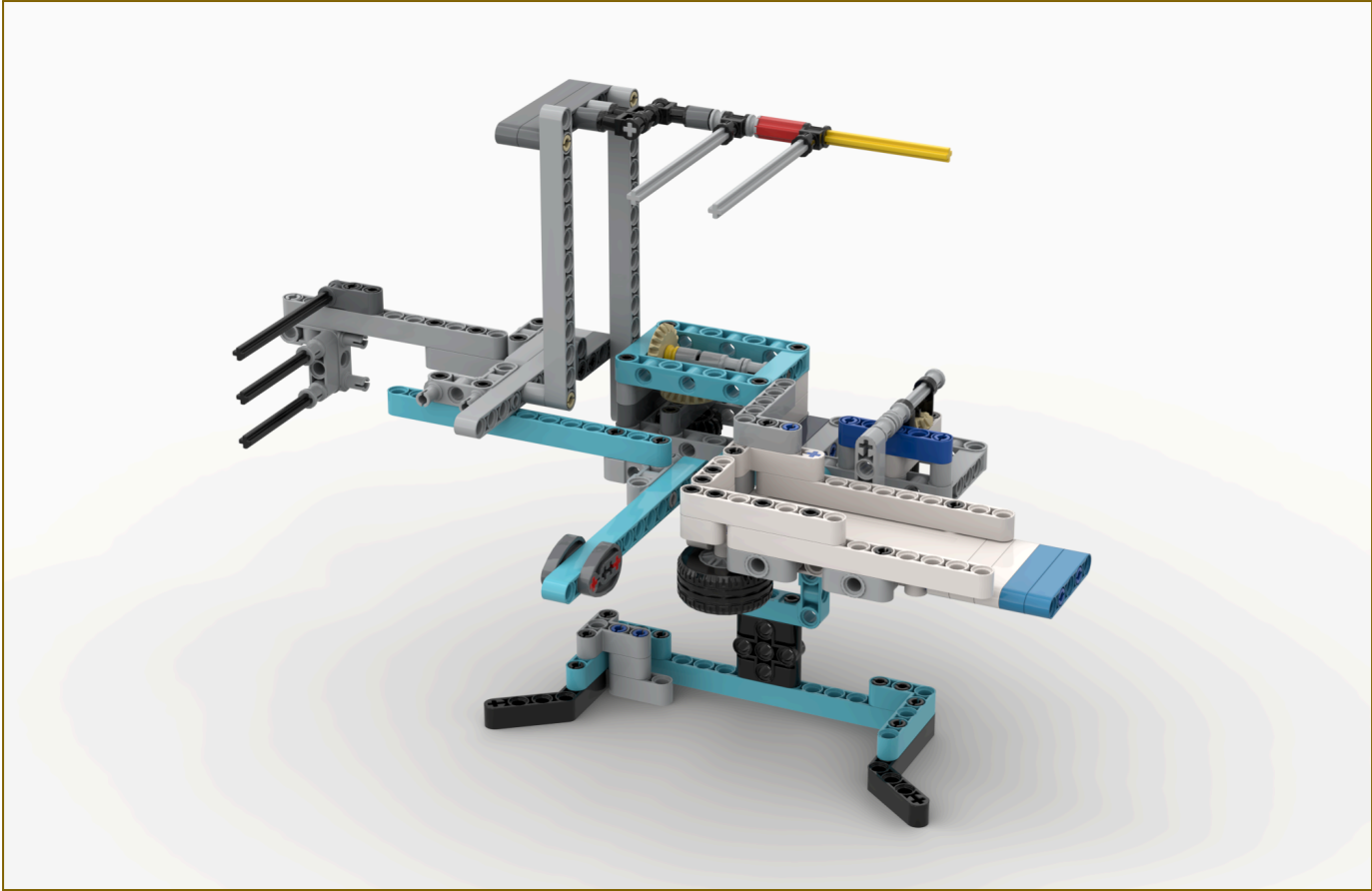
Sortie C



Mission	Pointage	Type de mécanisme
M03 - Récif corallien	20 pts	Actif
M02 - Plongeur sous-marin	20 pts	Actif
M14 - Collecte d'échantillons	10 pts	Actif
M02 - Requin	10 pts	Actif
M15 - Navire de recherche	15 pts	Indépendant
Total	75 pts	

Temps
23 secondes

Mécanisme - Couteau Suisse

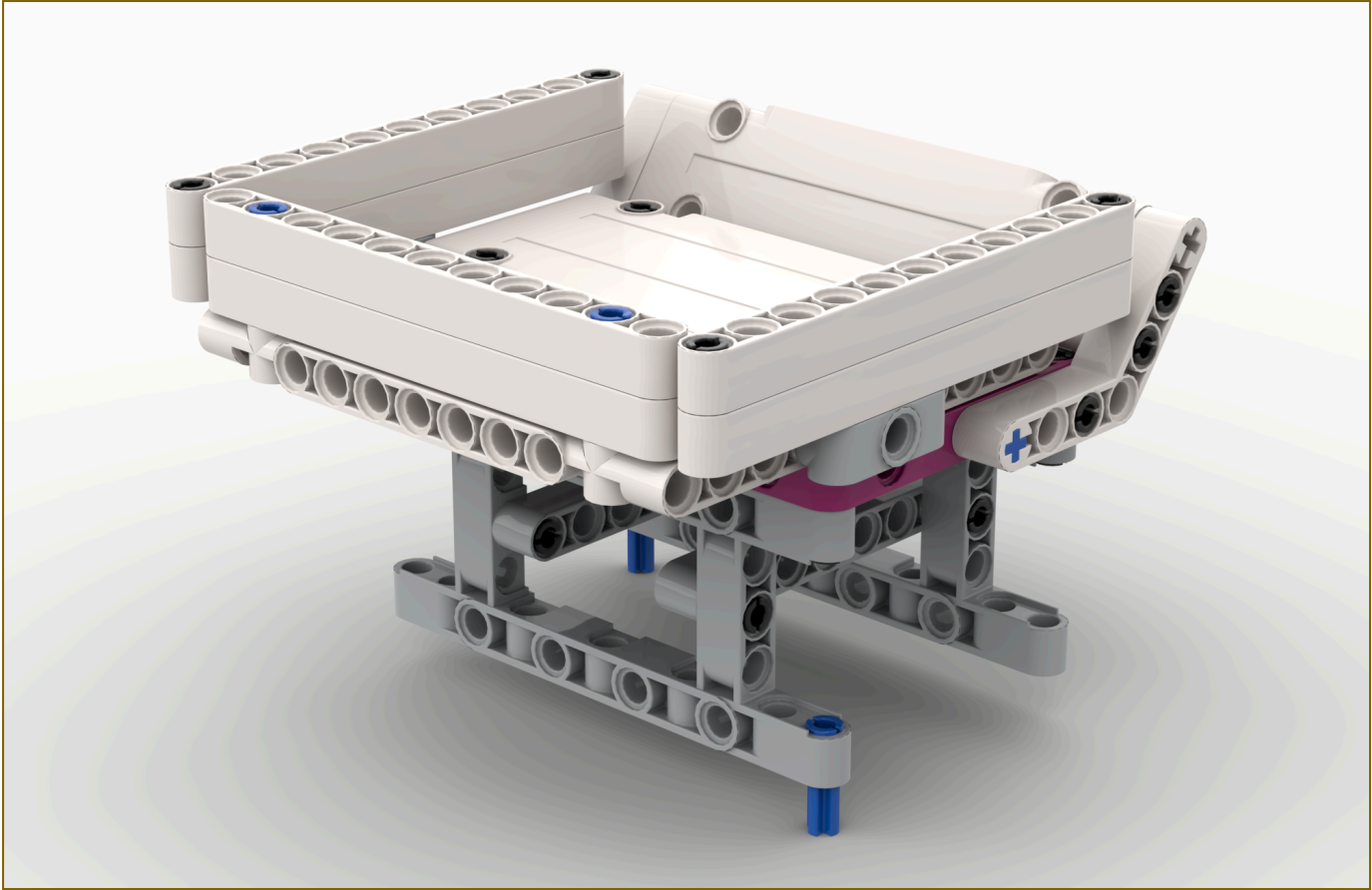


Type	Actif
Position	Avant

Mécanisme avec engrenage	Ratio	Effet
Bras avants	1.67:1	La vitesse est diminuée de 1.67:1 fois. La force est augmentée de 1.67:1 fois.
Glissade du requin	1:1.67	La vitesse est augmentée de 1.67:1 fois. La force est diminué de 1.67:1 fois.

Problème rencontré	Solution apportée
La glissade était trop petite donc le requin tombait	Nous avons allongé la glissade.
Le marteau n'était pas assez fort.	Nous avons ajusté la vitesse du marteau pour qu'il ait plus de puissance de frappe
La fourchette ne levait pas assez haut pour retirer l'échantillon de sol	Nous avons fait tourner le robot au lieux de le monter

Mécanisme - Bocal

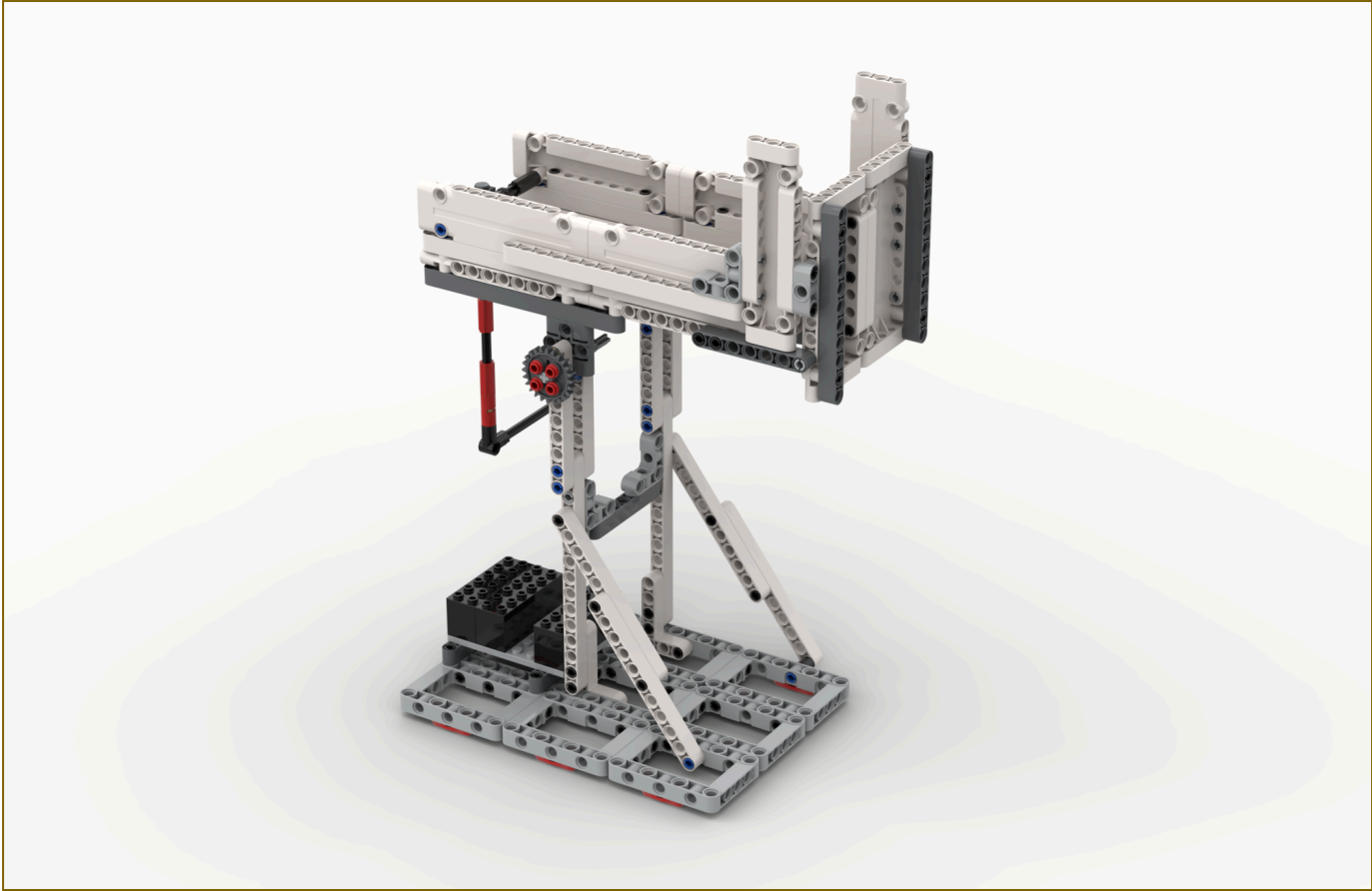


Type	Passif
Position	Sur le dessus du robot

Mécanisme avec engrenage	Ratio	Effet
Aucun	-	-

Problème rencontré	Solution apportée
Le bouton n’était pas accessible	Nous avons monté le bocal
Les éléments tombaient facilement	Nous avons augmenté les parois

Mécanisme - La chute



Type	Indépendant
Position	Dans la zone de départ

Mécanisme avec engrenage	Ratio	Effet
-	-	-

Problème rencontré	Solution apportée
Les pièces tombaient trop sur le côté	Nous avons ajouter une barre pour diriger les pièces
Le support de la chute n'était pas assez solide	Nous avons rajouté des supports pour le solidifier
La barre restait pris dans le robot	Donc nous avons fait en sorte que la barre puisse tourner facilement.

Programmation

```
def sortieC():
    moteur_droit = Motor(Port.E, positive_direction=Direction.CLOCKWISE, gears=[12,20], reset_angle=True)
    moteur_gauche = Motor(Port.A, positive_direction=Direction.CLOCKWISE, gears=[20,12], reset_angle=True)

    moteur_droit.control.limits(900,4000,200)

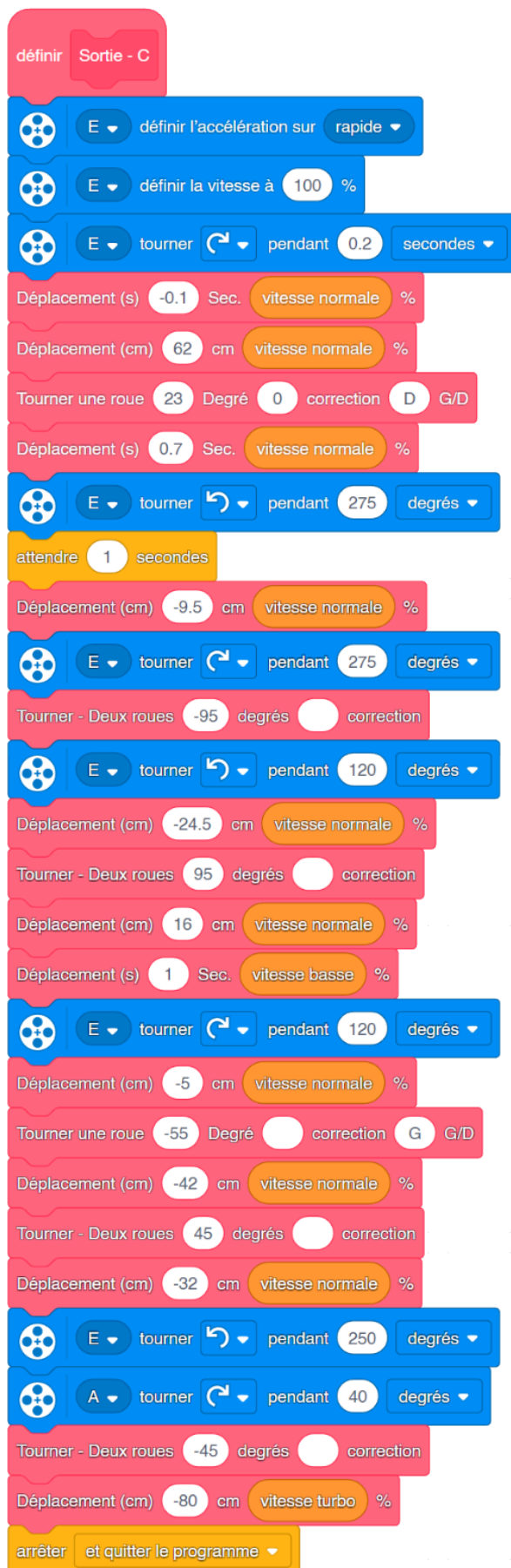
    #On initialise le robot
    moteur_droit.run_angle(180,15, wait=False)
    deplacerTemps(-100,vitesse_normale)

    #M03 - Récif corallien & M04 - Plongeur sous-marin
    deplacerDistance(670,vitesse_normale)
    tournerUneRoue(23,"droite")
    deplacerTemps(1000, vitesse_normale)

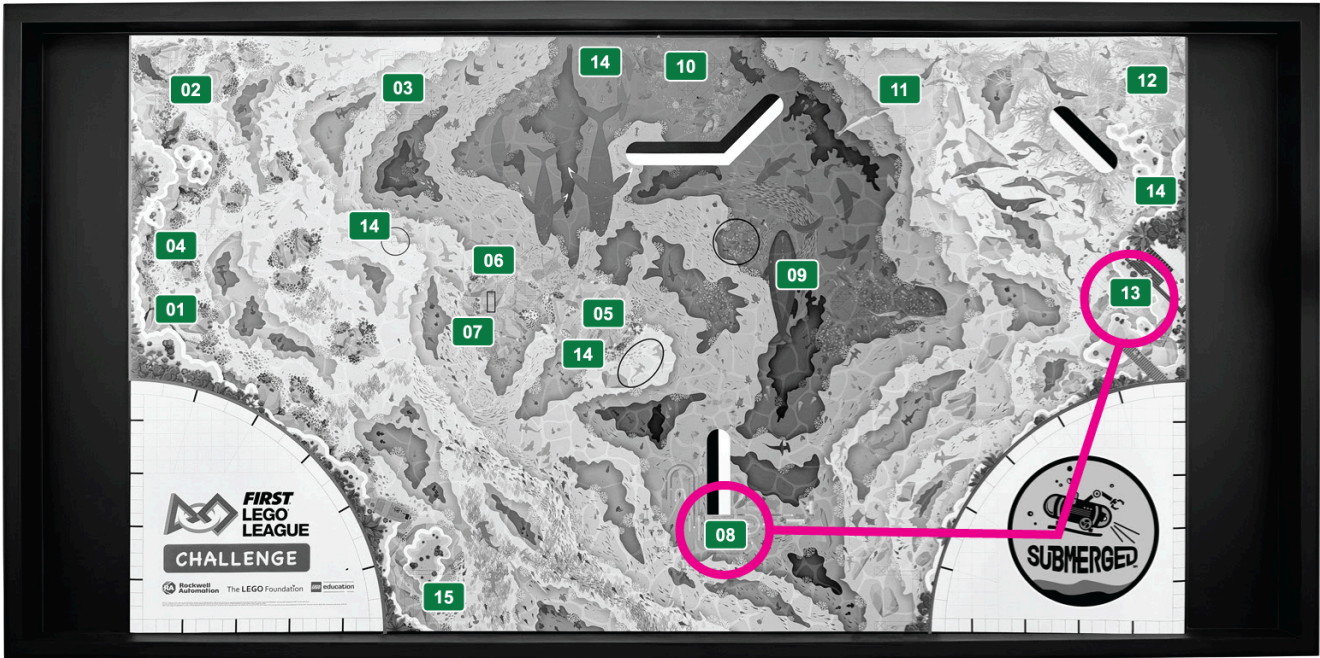
    moteur_droit.run_angle(1000,-135)
    wait(1000)

    #M14 - Collecte d'échantillons (Fond)
    deplacerDistance(-105,vitesse_normale)
    moteur_droit.run_angle(1000,135, wait=False)
    tournerDeuxRoues(-100)
    moteur_droit.run_angle(180, -70, wait=False)
    deplacerDistance(-265, vitesse_normale)
    tournerDeuxRoues(100)
    deplacerDistance(160, vitesse_normale)
    deplacerTemps(600, vitesse_basse)
    moteur_droit.run_angle(180, 90)
    deplacerDistance(-50, vitesse_normale)
    tournerUneRoue(-60, "gauche")

    #M02 - Requin
    deplacerDistance(-435, vitesse_normale)
    tournerDeuxRoues(50)
    deplacerDistance(-340, vitesse_normale)
    moteur_droit.run_angle(600, -120, wait=False)
    moteur_gauche.run_angle(300, 80)
    wait(300)
    tournerDeuxRoues(-56)
    deplacerDistance(-800, vitesse_elevee)
```

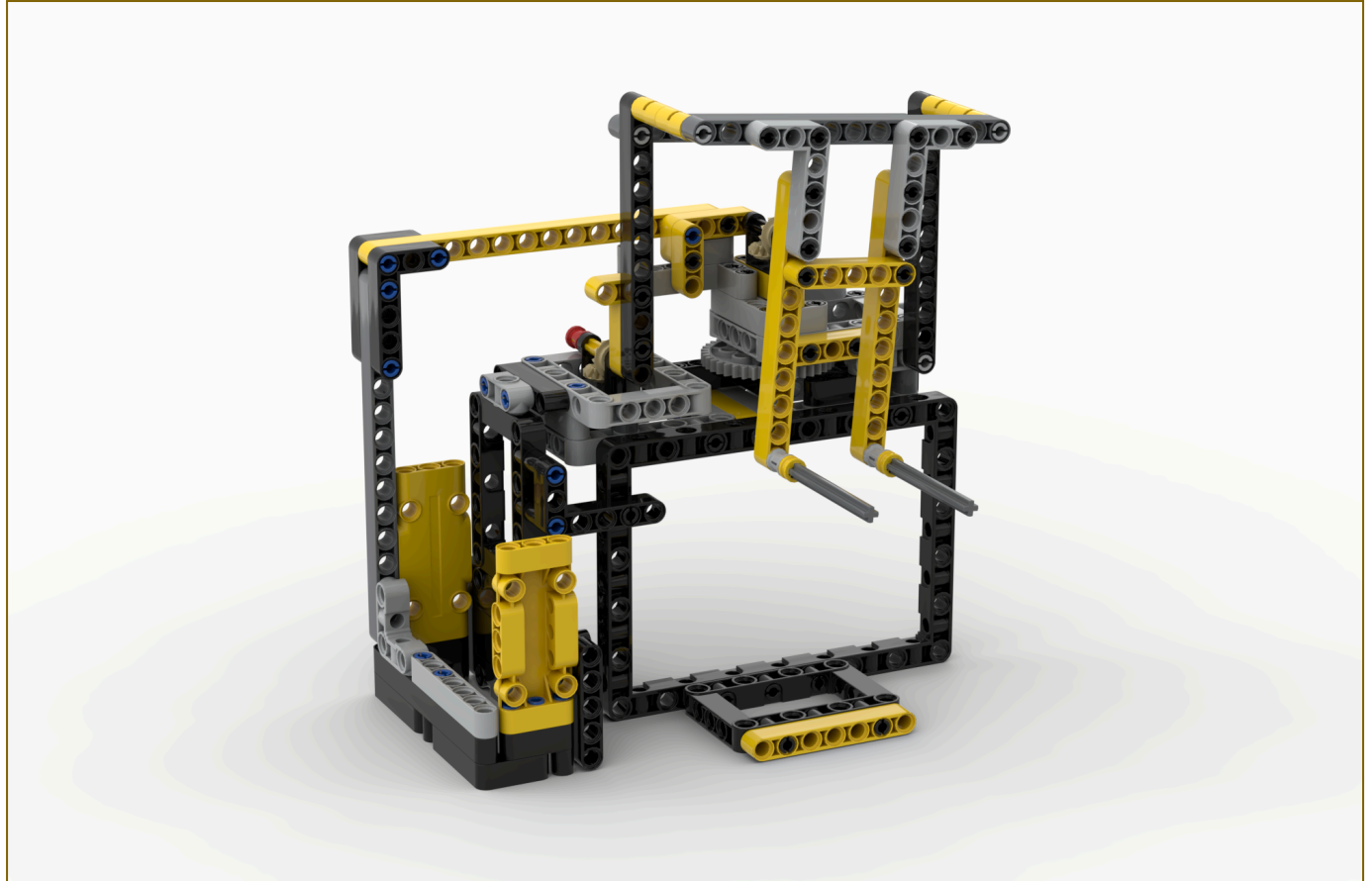
tSortie D



Mission	Pointage	Type de mécanisme
M13 - Modifiez les couloirs de navigation	20 pts	Actif
M09 - Habitat artificiel	20 pts	Passif
M03 - Récif corallien	15 pts	Passif
Total	55 pts	

Temps
12 secondes

Mécanisme - Fourche



Type	Actif
Position	Devant du robot

Mécanisme avec engrenage	Ratio	Effet
Tourner les tiges	1.67:1	Force augmentée
Libérer les corails	2.99:1	Force augmentée

Problème rencontré	Solution apportée
Le bateau tournait mal.	On a trouvé des tiges de bonne longueur.
La force des engrenages n'était pas suffisante.	On a fait un ratio plus puissant.
Les coraux tombaient.	On a mis des murs plus hauts.
Le module n'était pas solide.	On a solidifié avec une barre de 5.

Programmation

```
def sortieD():

    moteur_droit = Motor(Port.E, positive_direction=Direction.CLOCKWISE, gears=[12,20], reset_angle=True)
    moteur_gauche = Motor(Port.A, positive_direction=Direction.COUNTERCLOCKWISE, gears=[[20,12],[10,40]],
reset_angle=True)

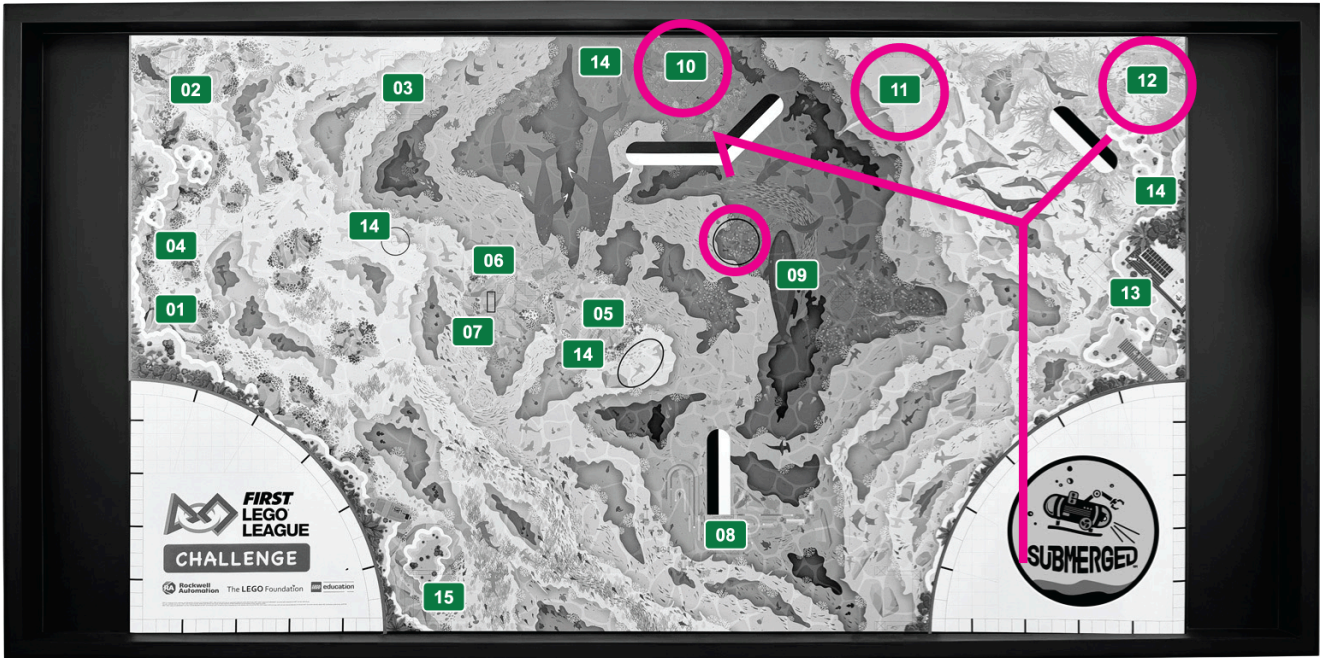
    #On initialise Le robot
    deplacerTemps(-200,vitesse_normale)

    #M13 - Modifier Les couloirs de navigation et M03 - Récif corallien
    deplacerDistance(400, vitesse_normale)
    tournerDeuxRoues(64)
    deplacerTemps(500, vitesse_basse)
    moteur_droit.run_angle(240,170,wait=False)
    moteur_gauche.run_angle(60,-90)
    wait(100)

    #M09 - Rencontre inattendue
    deplacerDistance(-220, vitesse_tres_elevee)
    moteur_gauche.run_angle(300,90,wait=False)
    tournerDeuxRoues(85)
    deplacerTemps(-2000,vitesse_basse)
    deplacerDistance(600,vitesse_elevee)
```



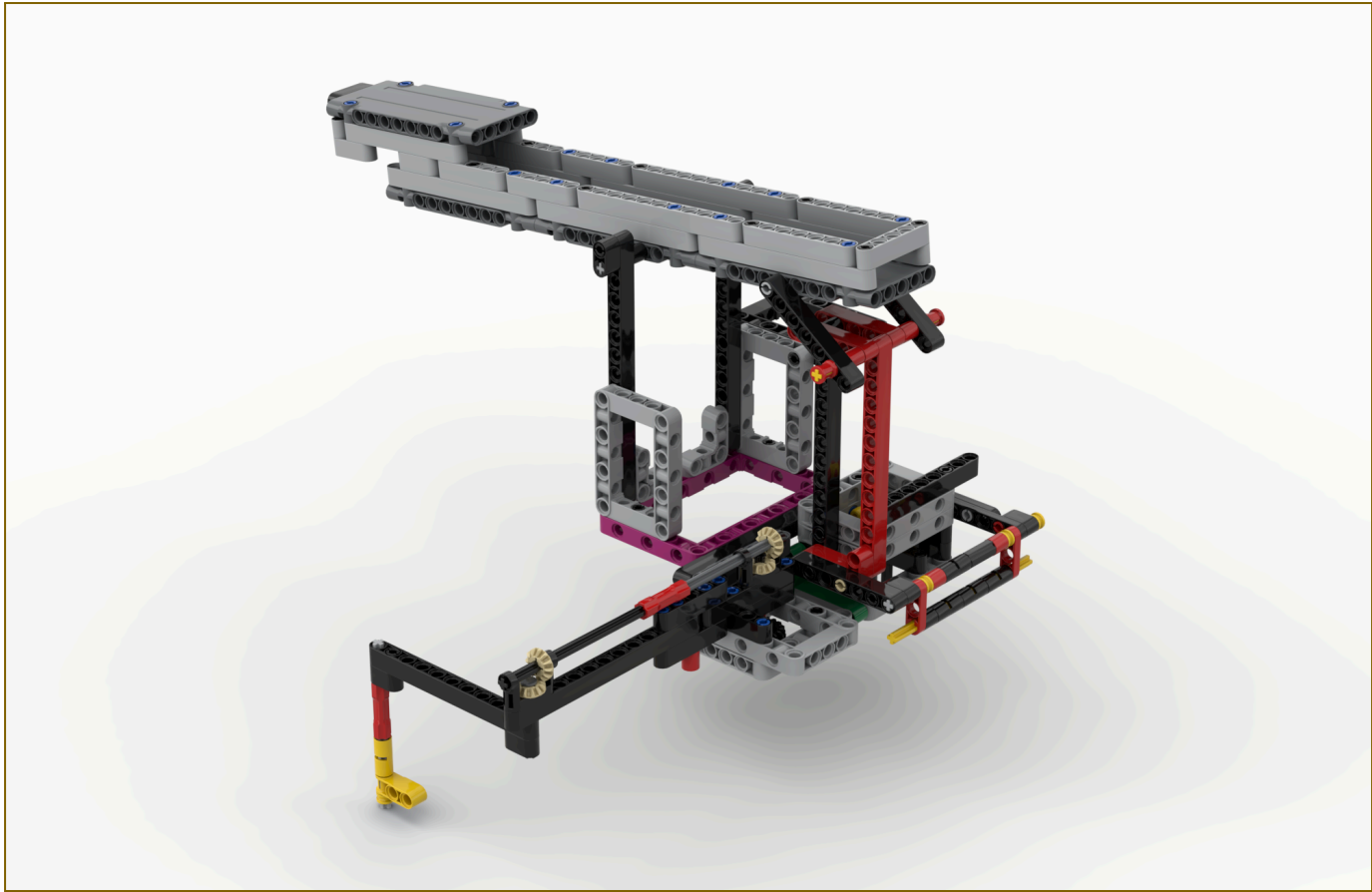
Sortie E



Mission	Pointage	Type de mécanisme
M12 - Nourrissez la baleine	40 pts	Passif et actif
M11 - Découverte du sonar	30 pts	Actif
M10 - Envoyez le sous-marin	40 pts	Actif
M09 - Rencontre inattendue	10 pts	Passif
Total	120 pts	

Temps
17 secondes

Mécanisme - Glissade

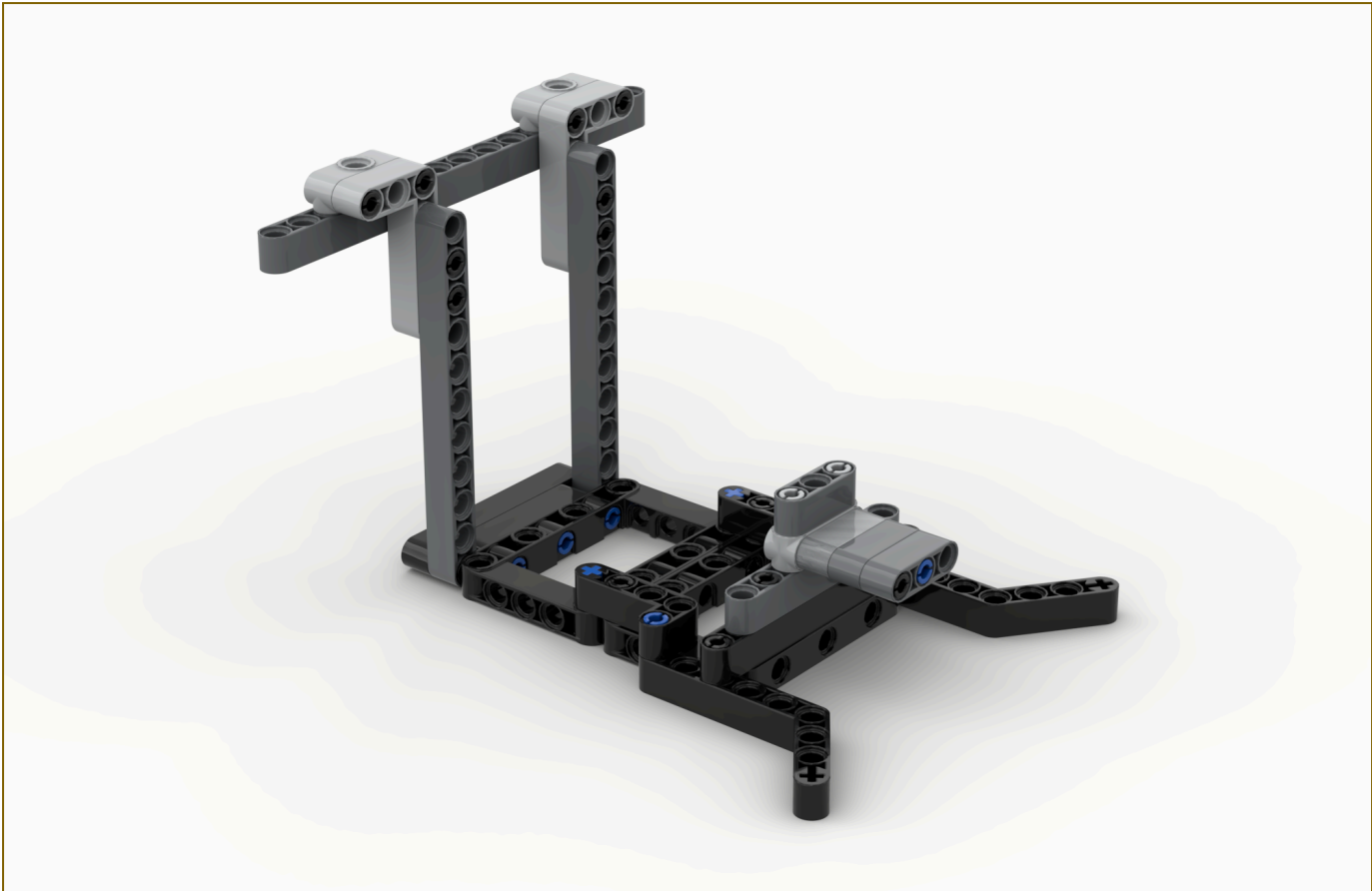


Type	Actif
Position	Devant

Mécanisme avec engrenage	Ratio	Effet
Mixeur	1.67:1	La vitesse est diminuée de 1.67:1 fois. La force est augmentée de 1.67:1 fois.
Levier	1:1.67	La vitesse est augmentée de 1.67:1 fois. La force est diminué de 1.67:1 fois.

Problème rencontré	Solution apportée
Le mixeur ratait sa cible souvent	Nous avons ajouté une palette.
Les krills tombaient à l’extérieur parfois	Nous avons ajouté un stoppeur au bout.

Mécanisme - Guide



Type	Passif
Position	Derrière

Mécanisme avec engrenage	Ratio	Effet
Aucun	-	

Problème rencontré	Solution apportée
La baleine tombait quelquefois	Nous avons ajusté la tige.

Programmation

```
def sortieE():
    moteur_droit = Motor(Port.E, positive_direction=Direction.CLOCKWISE, gears=[12,20], reset_angle=True)
    moteur_gauche = Motor(Port.A, positive_direction=Direction.CLOCKWISE, gears=[20,12], reset_angle=True)

    #On initialise le robot
    deplacerTemps(100,vitesse_basse)
    moteur_gauche.run_angle(200,-20, wait=False)

    #M12 - Nourrissez la baleine
    deplacerDistance(-650, vitesse_normale)
    tournerDeuxRoues(60)
    deplacerDistance(-230, vitesse_normale)
    deplacerTemps(-200, vitesse_normale)
    moteur_gauche.run_angle(300,130)
    wait(1000)
    moteur_gauche.run_angle(360,-130, wait=False)

    #M11 - Découverte du sonar
    deplacerDistance(130,vitesse_basse)
    tournerDeuxRoues(48)
    deplacerDistance(205,vitesse_normale)
    moteur_droit.run_angle(360,550)

    #M10 - Envoyez le sous-marin
    moteur_droit.run_angle(360,-180, wait=False)
    deplacerDistance(180,vitesse_normale)
    tournerDeuxRoues(44)
    deplacerDistance(180,vitesse_normale)
    deplacerTemps(600,vitesse_basse)
    moteur_gauche.run_angle(300,130)
    wait(1000)

    #M09 - Découverte inattendue
    tournerUneRoue(-20,"droit")
    deplacerDistance(-220,vitesse_normale)
```